NORTHWESTERN UNIVERSITY

INTERACTION OF DISCOURSE PLANNING, INSTRUCTIONAL PLANNING AND DIALOGUE MANAGEMENT IN AN INTERACTIVE TUTORING SYSTEM

A DISSERTATION

SUBMITTED TO THE GRADUATE SCHOOL
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

for the degree

DOCTOR OF PHILOSOPHY

Field of Computer Science

By

REVA K. FREEDMAN

EVANSTON, ILLINOIS

December 1996

© Copyright by Reva K. Freedman 1996

All Rights Reserved

Abstract

We demonstrate the utility of natural language generation as the underlying model for an intelligent tutoring system (ITS) in cardiovascular physiology. We have achieved this goal by dividing it into three subgoals, each of which builds on its predecessor: (a) developing a model of the tutorial dialogue of human tutors based on current research in natural language generation, with emphasis on text planning and the Conversation Analysis school, (b) analyzing a corpus of human-to-human tutoring sessions in cardiovascular physiology in terms of this model, and (c) designing an ITS which implements the model. We develop an abstract model of tutorial dialogue in order to put text generation for ITSs on solid theoretical footing. We give a detailed analysis of our corpus using this model, including a discussion of how tutors sequence their corrections, begin and end phases of the discourse, acknowledge responses, reply to student errors, teach different kinds of information, provide hints, conduct interactive explanations and choose between domain models. We present a detailed design for an ITS which uses this model to show that it can be implemented with current technology. The system is divided into two routines running in parallel, a global tutorial planner, which makes discourse decisions for units larger than a turn, and a turn planner, which assembles individual turns. The tutorial planner does not generate text directly, but generates a series of semantic forms. The turn planner collects the semantic forms for a turn, which may include information from multiple tutorial and/or conversational goals, and generates text for them as a unit. This architecture promotes coherent dialogue while permitting the tutor to use multi-turn discourse plans and change plans in response to student input. We expect this model to produce longer, more complex, and more varied dialogues than previous work.

Acknowledgments

When I began graduate school, I did not understand why dissertations always have long acknowledgments. Now I do. During my long career as a graduate student, many people have gone out of their way to be helpful. Several of them deserve special mention here. My advisor, Professor Gilbert K. Krulee, was always ready with the right mixture of advice, support, and action. Professor Martha W. Evens provided light at the end of the tunnel at a crucial moment. Leah Miller provided moral support on many occasions during our shared time at Northwestern. My parents, Eli and Miriam Freedman, have had unwavering faith in me. Finally, Michael Glass has supported this effort in every way possible, both personal and professional, from the beginning.

I benefited greatly from the opportunity to test my theories about text generation in the context of a real application. I am grateful to Professor Martha W. Evens of the Illinois Institute of Technology for making this opportunity available to me as part of the CIRCSIM-Tutor project. This portion of the work could not have been completed without the assistance of Professors Allen A. Rovick and Joel A. Michael of Rush Medical College, co-investigators and domain experts. The CIRCSIM-Tutor project came into being because they are dedicated teachers who really care that their students learn the fundamentals of cardiovascular physiology. This dedication led them to become interested in CAI systems long before most of their colleagues. They also designed the data collection experiments and served as tutors in those experiments. I would also like to thank all of the students, past and present, who have worked on the CIRCSIM-Tutor project for providing a rich environment for further research.

R. Michael Young generously permitted the CIRCSIM-Tutor project to use an early version of the LONGBOW discourse planner. Michael Elhadad not only built the FUF text

realization system, along with the SURGE grammar which accompanies it, but has continued to improve it and make it available to the text generation community.

Tom Dibble, Jay Freedman (no relation) and Chris Rickman, three people whom I have only met online, provided software to coerce a well-known word-processing package, whose name will not be mentioned here, into formatting a dissertation. Michael Glass also contributed software for this purpose.

Part of this work was supported by the Cognitive Science Program of the Office of Naval Research under Grant No. N00014–94–1–0338. The content does not reflect the position or policy of the government and no official endorsement should be inferred.

In memory of my beloved sister
SARA ELLEN FREEDMAN
May her memory serve as a blessing

Table of Contents

List of Figures

Introduction

> *It is impossible to dissociate language from science or science from language, because every natural science always involves three things: the sequence of phenomena on which the science is based; the abstract concepts which call these phenomena to mind; and the words in which the concepts are expressed. To call forth a concept a word is needed; to portray a phenomenon, a concept is needed. All three mirror one and the same reality.*
>
> —*Antoine Lavoisier*

The purpose of this dissertation is to demonstrate the utility of text generation as the underlying model for an intelligent tutoring system (ITS).

Our approach to this goal encompasses three subgoals:

- To develop a model of the dialogue of human tutors based on current research in natural language generation, with emphasis on text planning and the Conversation Analysis school.
- To analyze a corpus of human-to-human tutoring sessions in cardiovascular physiology in terms of this model.
- To design a large-scale, practical ITS which implements the model.

In addition to demonstrating the importance of text generation in the design of intelligent tutoring systems, we hope that the system described here will prove useful to students beginning the study of cardiovascular physiology.

The result of this research is a design for a new planner called TIPS[1] which

[1] TIPS stands for "Text generation Interactively, a Planning System." We chose this acronym because of the pun involved: one of the system's primary pedagogical goals is to generate hints for the student, i.e. tips.

incorporates both discourse planning and instructional planning. TIPS uses text planning as an underlying model, rather than the more common approach of considering text generation as an afterthought to pedagogical planning. As all decisions, even pedagogical ones, must eventually be expressed as text, TIPS views responding to the student in an ITS as a text planning problem. Instead of a top-level goal of "guide the student to the correct value of X," our top-level goal is "generate text in which the tutor guides the student to the correct value of X."

TIPS will be used as the planner for v. 3 of CIRCSIM-Tutor, an ITS currently under development at the Illinois Institute of Technology and Rush Medical College. CIRCSIM-Tutor[2] tutors students on the baroreceptor reflex, the negative feedback system which maintains a steady blood pressure in the human body. The baroreceptor reflex is one of the more difficult topics in the first-year medical curriculum. The goal of CIRCSIM-Tutor is to help students integrate the material which they have learned so that they can use it effectively.

The term "intelligent tutoring system" has been applied to a wide variety of systems with goals including coaching, teaching, testing, assisting student exploration, and providing after-the-fact analysis. User interfaces vary from simulation-based graphics to systems which are largely text-based. Although the optimal choice of output medium for an ITS depends on the subject matter and goals of the system, the use of natural language usually permits the tutor to present more detailed information to the student than other modalities.

Generating text from a knowledge base has several potential advantages over the use of canned text:

---

[2] Unless otherwise specified, the term CIRCSIM-Tutor refers to v. 3.

- It frees us from dependency on a small list of precomputed problems.
- It allows us to have a variety of ways to say the same thing without having to code each one explicitly.
- It increases the coherence of the dialogue by allowing us to plan text as a unit.
- It lets us respond specifically to issues which the student brings up.

However, these potential advantages have often not been realized for both theoretical and practical reasons. Designers of text-based ITSs have faced a difficult choice from the inception of the genre by J. R. Carbonell [1970]. Since the ability to generate text from concepts has not developed as rapidly as the set of concepts which need to be expressed, authors have had to choose between natural-sounding text which has been largely hand-crafted and the simplified, mechanical-sounding text which can be generated by methods such as the use of surface-structure templates. Using hand-crafted text restricts the practical size of a tutoring system by reducing or eliminating the benefits which come from economy of scale. On the other hand, the use of superficial methods of text generation reduces the cohesiveness and precision of the resulting text, an equally undesirable result.

The emphasis in CIRCSIM-Tutor is on tutoring students on material which they have already studied, not on teaching new material. This orientation leads us to place greater emphasis on the interactive aspects of the conversation, rather than on elaborate ways to present material. Thus the following criteria are important to us:

- Generated text must be organized into cohesive turns containing text of appropriate complexity for the material.
- Text must flow in natural conversation patterns. In other words, the transitions between turns must approximate human patterns.
- A variety of pedagogical methods must be used.

- Syntactic structures and lexical items must not be repeated overly often.

We believe that the quality of the language directly affects students' interest, understanding and retention. Variety and coherence are two of the most important factors influencing students' perception of the quality of the language. When each sentence generated by an ITS makes sense but the resulting text is difficult to follow, coherence is generally the issue. At the pedagogical level, a variety of methods must be available if the tutor is to be able to help the greatest number of students. At the linguistic level, variety is important to ensure that students continue to read the tutor's output and do not resort to solving the problems by rote.

These considerations have led us to choose the following goals for the TIPS system:

Planning:

1) To maintain a dialogue, including appropriate responses to student initiatives and errors, while carrying out a global tutoring plan.

2) To provide the ability to use multi-turn plans and drop or amend them if the student does not respond as expected.

Tutorial:

3) To provide the ability to teach the same concept in multiple ways.

4) To provide the ability to provide specific content-based responses to common student errors.

5) To generate tutorial discourse phenomena similar to the hints, explanations and interactive sequences generated by expert human tutors.

Discourse:

6) To produce natural-sounding dialogue over a dialogue of arbitrary length, i.e. inter-turn coherence.

7) To produce natural-sounding turns, i.e. intra-turn coherence, and appropriate use of pronouns and subordinate clauses.

8) To provide the ability to say the same thing in more than one way.

Practical:

9) To allow for principled expansion at all levels, including discourse schemata, semantic concepts, surface structures and lexical items.

10) To maintain a separation of functions enabling the easy addition of multi-lingual output in the future.

11) To provide sufficiently fast response time for regular student use.

The fundamental division of labor in the TIPS system is between the *tutorial planner*, which makes discourse decisions for units larger than a turn, and the *turn planner*, which assembles individual turns. These modules run in parallel to keep track of pedagogical goals and the evolving conversation. The tutorial planner develops a global, top-down plan for a tutoring session, which it can update at every turn. For efficiency, this plan is maintained at a high level and expanded only when necessary. The tutorial planner uses tutoring goals stored in a schema-based format, which it augments on a turn-by-turn basis in order to respond to student utterances. Plan operators in the TIPS library cover pieces of text which range in size from the whole conversation down to primitive speech acts. Constraints on the plan operators can be used to take other knowledge sources into account, such as the domain knowledge base, the pedagogical knowledge base, and the student model. Although TIPS uses multi-turn schemata, it is responsive to the student because it is capable of replanning at every turn, dropping an unproductive schema in favor of a different one.

The turn planner is responsible for combining semantic forms emitted by the tutorial planner into turns, and ensuring intra-turn and inter-turn cohesion. It accumulates

semantic forms until it receives a goal which requires a response from the student. At that point it combines the semantic forms into a cohesive turn, attaches that turn to the discourse tree which it maintains, and issues the text to the student.

We believe that this design will permit the generation of longer and more complex dialogues than previous work. This dissertation concentrates on three aspects of the design: demonstrating the feasibility of the architecture, illustrating how it can be used to satisfy our pedagogical goals, and outlining the tutorial knowledge base needed for tutoring real problems. In order to provide broad coverage of the subject matter and of the methods used by our expert tutors, we have omitted some of the details necessary for a working system.

CIRCSIM-Tutor is an ITS which generates text about solving problems. To describe such a system, it is necessary to describe the following aspects:

- How to solve the problem and represent the solution (Chapter 2)
- How to teach the solution and represent the pedagogy (Chapters 3 and 4)
- How to plan a conversation and generate text (Chapter 5)

This dissertation is organized as follows:

Chapter 1 takes a critical look at the ITS literature to identify relevant factors in the design of an ITS. We analyze the relation of tutorial, discourse and linguistic knowledge in a selection of well-known ITSs. Then we review some key systems from the planning and text generation literature. We compare CIRCSIM-Tutor to the best examples of natural-language based explanation systems. Finally, we study some ITSs which have made creative use of causal models.

Chapter 2 gives an overview of the baroreceptor reflex domain. It describes the domain model and presents rule-based methods for solving problems in the domain.

Appendix A contains a complete solution trace for every simple problem in the domain as well as a representative sample of more complex problems.

Chapter 3 introduces the CIRCSIM-Tutor project. In addition to a description of the user interface and the student protocol, it contains a brief history of earlier versions of CIRCSIM-Tutor and of previous CAI systems for this domain.

Chapter 4 describes a model of tutoring discourse derived from the study of naturalistic data from expert tutors. We develop an abstract model of tutorial dialogue. We give a detailed analysis of our corpus using this model, including a discussion of how tutors sequence their corrections, begin and end phases of the discourse, acknowledge responses, reply to student errors, teach different kinds of information, provide hints, conduct interactive explanations and choose between domain models. Although we have tried to make the examples comprehensible without any previous knowledge of physiology, this has not always been possible. We close with a look at some of the problems of cooperative conversation illustrated in the transcripts.

Chapter 5 describes the design of the TIPS planner. We describe the design of the tutorial planner, its relation to the turn planner, and the motivation for the use of schemata as the central form of representation for tutorial knowledge. We illustrate the dynamic behavior of the planner, showing its behavior both in normal and exceptional conditions. Then we briefly describe the functions of the turn planner, including the combination of semantic forms into sentences, the lexicalization and linearization processes, and the building of the discourse tree.

Chapter 6 contains our conclusions. We describe some advantages of the system proposed here. We outline some proposed enhancements to the current architecture, including the derivation of more sophisticated explanations through the use of a functional

model of the domain. We explain why other proposed enhancements are not feasible using this architecture. Finally, we describe potential applications of the TIPS planner to text generation outside of ITSs.

Chapter 1
Issues in the Design of Text-Based Intelligent Tutoring Systems

Education is not the filling of a pail, but the lighting of a fire. —*W. B. Yeats*

The term "intelligent tutoring system" (ITS) has been applied to a wide variety of systems with goals ranging from teaching and testing to providing after-the-fact analysis, coaching, and assisting student exploration. User interfaces vary from simulation-based graphics to systems which are largely text-based. As a natural-language based ITS, CIRCSIM-Tutor draws from several areas of computer science, including planning, text generation and knowledge representation. In this chapter we situate CIRCSIM-Tutor in its theoretical and historical context in order to motivate the design goals given in the Introduction.

We start by defining a set of axes for classifying ITSs to show how CIRCSIM-Tutor fits into the ITS world. Then, since a key fact about CIRCSIM-Tutor is that it separates domain knowledge (what to teach) from tutorial knowledge (how to teach it), and tutorial knowledge from discourse knowledge (how we express pedagogical ideas in words), we describe several historical ITSs to survey the evolution of interest in these concepts.

A second key fact about CIRCSIM-Tutor is that it uses a global planner and text generation methodology in order to generate more cohesive conversations. So we survey some of the fundamental research in planning and text generation. We review some of the more important historical systems and some current large-scale explanation systems in order to show how CIRCSIM-Tutor fits in with respect to other natural-language based

systems.

Finally, CIRCSIM-Tutor domain knowledge includes causal reasoning. Following Stevens, Collins and Goldin [1979], we use the term *causal* for relationships which involve causation in the physical world and *functional* for dependencies which involve only a logical notion of causation, such as those involving a definition or role relationship. We end this chapter with a description of some important ITSs which made innovative use of the ideas of causal and functional modeling.

1.1 Relation of domain, tutorial and discourse knowledge in ITSs

1.1.1 Criteria for classifying ITSs

Different researchers have studied widely different parts of ITSs. The following chart shows some of the categories which earlier researchers have found relevant, along with a number of options for each. Many of these issues are related. In this dissertation we will touch on each of these issues except for input understanding and the student model, which are outside the scope of this research. In the remainder of this section, we will show how problems in some of these areas have led later researchers toward more sophisticated designs.

1. Representation of domain knowledge
   a) Facts only
   b) Quantitative model
   c) Qualitative model for script-style causal reasoning
   c) Causal reasoning including functional model

2. Separation of types of knowledge
   a) No separation of types of knowledge
   b) Separation of domain and pedagogical/discourse knowledge
   c) Separation of discourse and pedagogical/domain knowledge
   d) Separation of domain, pedagogical and discourse knowledge

3. Representation of student model
   a) Same representation as knowledge to be taught, e.g. like the CIRCSIM-Tutor prediction table
   b) More abstract derived representation, e.g. including student's underlying misconceptions
   c) Personalized student model, i.e. including answers to questions asked for diagnosis or confirmation of diagnosis

4. Making sense of student input
   a) Keyword matching
   b) Parsing (any form)
   c) Parsing with recognition of ill-formed input
   d) Goal recognition
   e) Plan recognition

5. Type of tutoring transactions
   a) Teacher-led
      1) Expository teaching
      2) Socratic teaching (indirect)
      3) Problem-solving guidance
   b) Student-led
      1) Free discovery: tutor provides answers only
      2) Tutor provides intermittent feedback
      3) Tutor provides after-the-fact feedback

6. Inter-turn text planning
   a) Planning each turn separately
   b) Multi-turn plans
   c) Planning a turn with reference to previous turns

7. Intra-turn text planning and realization
   a) Neither pedagogical nor linguistic abstraction (template filling)
   b) Pedagogical abstraction only (multiple potential templates)
   c) Linguistic abstraction, e.g. intermediate semantic layer

8. Coherence of the conversation
   a) Coherence not explicitly considered
   b) Coherence from nature of task
   c) Local coherence from low-level planning
   d) Coherence from precompiled plan
   e) High-level planning isomorphic to pedagogical tasks
   f) High-level planning of general conversation

1.1.2 ITSs vs. traditional CAI systems

Traditional computer-aided instruction (CAI) systems usually consist of instructional units, often called frames, which contain text and questions. Usually the text is stored in surface form, and the domain knowledge is hard-coded as part of the text. Each frame contains pointers to all possible successors; the one to be used is a function of the student's input. Such systems are tedious to write because every possible student response must be accounted for at each step. In other words, there is no way to reason about answers in a general way. There is usually no way to reason about pedagogical knowledge either. In other words, there is no way to choose between multiple ways to teach a concept without hard-coding the choice. Finally, since the text is hard-coded, there is no way to provide multiple rhetorical patterns for expressing a teaching method or multiple syntactic forms for expressing a concept without coding new frames for each set of choices.

Such systems are effectively large finite-state machines which resemble the programmed textbooks popular in the 1950's and 1960's. It is this aspect which is essential to the characterization of CAI systems. For example, CIRCSIM (Section 3.1.3) tutors students on material they have studied earlier. As a result it does not alternate between text and questions in the way many people associate with frame-based CAI. Yet CIRCSIM is undoubtedly a CAI system: it can be viewed as a finite-state machine with over 200 states, each of which explains something to the student under a given set of conditions.

The characteristics of traditional CAI systems described above can be summarized as follows:

- Domain knowledge, including both knowledge about the solution to a problem and knowledge about how to solve the problem, is combined with pedagogical knowledge.
- Pedagogical knowledge—what to teach the student under which conditions—is not represented in a form which can be used for reasoning.
- Discourse knowledge, or how to choose a linguistic form for a pedagogical concept, is not represented separately from pedagogical knowledge.
- Linguistic knowledge, or how to express discourse knowledge as surface text, is not represented separately from discourse knowledge.

Since these issues are similar to the issues which motivated the development of other types of knowledge-based systems, it is natural to consider the use of artificial intelligence techniques to simplify the writing of instructional software. Artificial intelligence techniques can be used to reduce the amount of knowledge which must be specified through the use of suitable knowledge representations and mechanisms to manipulate them. For example, domain knowledge can be represented using as a semantic net, as a set of clauses with an inference mechanism, or using a variety of other mechanisms. Representing pedagogical knowledge as a set of plan operators with preconditions is one method for expressing the underlying concepts without needing to write out each case. Language is a special case. Since extensive work has been done in the area of natural language processing, many well-defined techniques are available which do not necessarily show their origins in AI.

Tutoring systems which use artificial intelligence techniques are called intelligent computer-aided instruction (ICAI) or intelligent tutoring systems (ITS). From this definition it can be seen that there is no clear-cut distinction between traditional CAI systems and ITSs, as there are several parts of a CAI system where AI techniques might

be useful and a variety of AI techniques which can be used.

1.1.3 Carbonell: Using AI in the domain model in SCHOLAR

SCHOLAR, developed by J. R. Carbonell [1970], is often considered in the United States to be the first ITS. It changed the paradigm for CAI by incorporating AI techniques into a tutoring system. SCHOLAR is a question-answering system based on a semantic network, using mainly the relations *is-a* and *is-part-of*. The knowledge covered includes declarative knowledge about rainfall; SCHOLAR does not teach any causal or procedural knowledge.

Parsing is simplified by restricting student answers to numbers, place names, and other types specified in the question. Errors are classified, so that, for example, a misspelled answer generates a different response than a completely wrong answer.

Carbonell called SCHOLAR a *mixed-initiative* system because it had two modes of operation: the teacher could ask the student questions or vice versa. Carbonell's purpose in doing this was to make the point that a more flexible program could meet the needs of a greater number of students. However, SCHOLAR could not hold a continuous conversation which required cooperative dialogue behavior. In fact, SCHOLAR does not attempt to create a coherent conversation.

SCHOLAR does not use global plans at either the tutorial or discourse levels. It had a prioritized list of topics to cover. In addition, it preferred previously-mentioned topics over new ones. SCHOLAR used several tutorial plans, including true/false questions, multiple-choice questions and short-answer questions. It chooses randomly among these plans. Text is generated from semantic forms using a simplified grammar without embedded clauses. Some concepts can be realized by a variety of surface forms. For example, the system can ask the student to answer "true or false" or "correct or

incorrect." The following short excerpt is typical of the SCHOLAR. system.

(1) T: The capital of Chile is Santiago. Correct or incorrect?
S: Correct.
T: That's right. Approx. what is the area of Brazil?
S: 2,500,000 square miles.
T: Wrong. Please indicate if the following statement is correct or incorrect: The area of Paraguay is approx. 47432 square miles.
(excerpted from Carbonell [1970, fig. 1])

In order to improve the output of SCHOLAR, Collins, Warnock and Passafiume [1975] studied dialogues with human teachers teaching geography. To create a structure for the conversation, their version used a depth-first search to determine the fact to be taught instead of a priority scheme. This change greatly increased the coherence of the generated dialogues. More importantly, this paper is the first of a series analyzing the productions of human tutors.

1.1.4 Collins & Stevens: Modeling Socratic dialogues

Since SCHOLAR used a semantic net to represent its domain knowledge, it could only handle factual knowledge. Carbonell and some of his colleagues, particularly Allan Collins, were interested in adding the capability to handle causal reasoning, especially the ability to do functional analysis. Functional analysis means the capacity to analyze the factors on which a hypothesis depends, e.g. the ability to infer a region's climate from its location. This ability would allow the tutor to teach the student not only the facts but the reasoning behind them. Collins et al. [1975] show how a desire to teach functional inference leads naturally to the study of the Socratic method.

In the Socratic method, also called the case, inquiry or discovery method, the teacher leads students to infer generalizations about the subject matter by asking a series of questions. Collins [1977] notes that the Socratic method is usually taught by example,

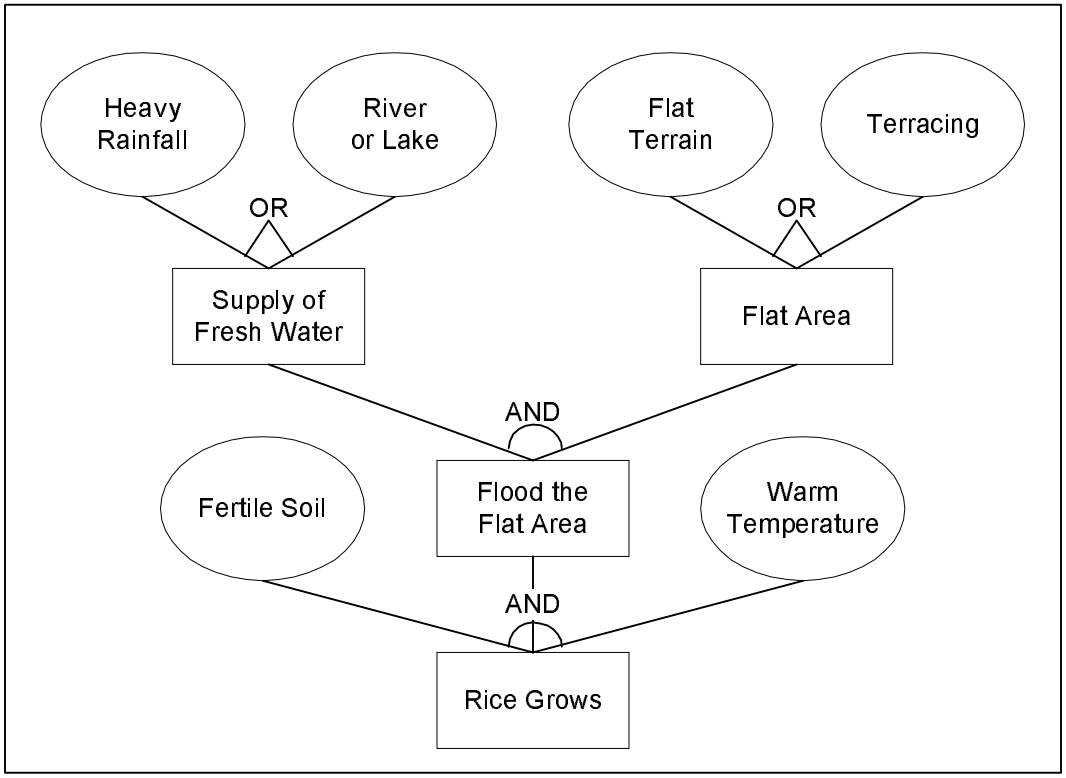

Figure 1.1: Factors affecting rice growing (from Collins [1977], fig. 1)

so that it is difficult to find a precise description of the method. To remedy that deficit, Collins did extensive research on dialogues conducted by himself and other teachers with the goal of identifying pedagogical strategies and the discourse strategies used to implement them.

Collins [1977] describes Socratic dialogues about causal factors affecting the growing of rice. These factors can be summarized in Figure 1.1. This chart shows four variables which can be combined to yield necessary and sufficient conditions for the growing of rice. Intermediate variables are shown along causal chains. For example, heavy rainfall is not necessary for the growing of rice because fresh water can be supplied by a river or lake. Collins gives a set containing two dozen production rules for conducting a

dialogue to teach the student about these factors. Here are two sample rules, along with examples, for teaching students not to make hasty generalizations.

> If the student gives a conclusion based on insufficient factors, inquire about the faulty generalization.
>
> > If the student gives water as the reason they grow rice in China, ask "Do you think any place with enough water can grow rice?".
>
> If the student gives a conclusion based on insufficient factors, give a counterexample.
>
> > If the student gives water as the reason they grow rice in China, ask a question like "Do they grow rice in Ireland?" or "Why don't they grow rice in Ireland?".

The complete set of rules includes cases where the student gives an extraneous factor or an incorrect answer. The rules can be combined to make dialogues such as the following:

(2)
T: Where in North America do you think rice might be grown?
S: Louisiana.
T: Why there?
S: Places where there is a lot of water. I think rice requires the ability to selectively flood fields.
T: O.K. Do you think there's a lot of rice in, say, Washington and Oregon?
S: Aha, I don't think so.
T: Why?

(excerpted from Collins [1977], table 1)

Collins also analyzes real and hypothetical dialogues on a variety of topics conducted by other teachers to show that the rules can be used in other domains. Although Collins does not explicitly give a hierarchical goal structure, all possible sequences for executing the rules lead to coherent text. Resnick [1977] attempts to produce a more directly executable version of Collins' rules by making the prerequisites more explicit.

Collins and Stevens [1982] give a finer-grained version of the rules in Collins [1977]. Additionally, they provide some data about potential high-level goals, based on think-aloud sessions with expert teachers. Finally, they consider the refinement of semantic

forms into more primitive forms. For example, they point out that each strategy can be implemented in three forms: questioning (*ask for*), suggesting (*suggest*) and commenting (*point out*). In the questioning form, which is the most frequently used, the teacher attempts to elicit information from the student. The suggesting form involves suggesting a factor for the student to consider. The commenting form involves giving information to the student.

Collins and Stevens [1991 (originally published in 1983)] analyze additional dialogues from a variety of fields and produce a list of *instructional techniques* shared by inquiry-oriented teachers. These techniques include selecting counterexamples, generating hypothetical cases, entrapping students ("so would you say that…") and tracing consequences to a contradiction.

Very little of Collins and Stevens' work was implemented due to the difficulty of understanding the student's responses and discerning which factors the student is referring to. Here is an excerpt from a sample dialogue from the WHY system [Stevens & Collins 1977]. The contrast between this dialogue and the elegant work of the human tutors may explain why further implementation work was not continued.

(3) T: Do you think the Amazon jungle has heavy rainfall or little rainfall?
S: Heavy rainfall.
T: Why does the Amazon jungle have heavy rainfall?
S: Because it's near the Atlantic.
T: Yes, the Amazon jungle is near the equatorial Atlantic. How does that affect the rainfall in the Amazon jungle?

(excerpted from Stevens and Collins [1977], table 1)

Recently, Wong, Looi and Quek [1996] have produced a prototype tutor based on the rules in Collins and Stevens [1982].

Collins' work is relevant to CIRCSIM-Tutor because it demonstrates the use of multiple tutorial plans for a goal. Thus a distinction has been made between domain

knowledge and the method of teaching it. The reference to questioning, suggesting and commenting forms is a step toward providing multiple discourse forms for a tutorial method. However, Collins does not make a distinction between a discourse form and potential syntactic implementations for it, as he does not discuss the linguistic implementation of his theories.

Among the specific methods discussed by Collins, some are relevant to CIRCSIM-Tutor, such as the idea of teaching by deriving a contradiction from the student's statements. Others, such as getting the student to state rules more carefully, do not apply because CIRCSIM-Tutor generally does not teach rules explicitly unless the student is having trouble. In particular, rules involving the comparison of two cases have no analogue in CIRCSIM-Tutor.

1.1.5 Clancey: Rule-based representation of tutorial knowledge in GUIDON

GUIDON [Clancey 1987] is a tutoring system which attempts to teach the student the problem-solving knowledge contained in the MYCIN expert system for medical diagnosis. As the student solves a problem, GUIDON tracks the student's progress against the solution trace provided by MYCIN. A difficulty intrinsic to GUIDON is that MYCIN is a rule-based system which does not use the abstractions and reasoning processes which medical students need to learn. This problem was ameliorated in later systems.

The tutorial rules in GUIDON contain knowledge about dialogue patterns. For example, one of the tutorial methods used by GUIDON is to go through each clause of a MYCIN rule and explain its status to the student. Another is to challenge the student's response by asking a question about the student's action. For example, GUIDON can use the following template to question one of the student's responses:

(4) T: Given <facts in premises>, does this tell you <your conclusion>?

GUIDON can provide more than one way to teach a concept. A set of alternative rules for the same goal is known as a *packet*.

GUIDON maintains coherence via a data structure called the *focus record*, a set of registers which keep track of the global context of the dialogue. In the absence of a global plan, the use of the focus record helps to ensure a certain amount of coherence. An additional measure of coherence is provided by the conversational structure induced by the program. Although GUIDON does not contain a global planner, by following MYCIN's solution, it generates a reasonably well-structured conversation. However, conversations with GUIDON can be difficult to follow, as evidenced by the following example:

(5)
T: What are the organisms that might be causing the infection?
S: Neisseria meningitidis.
T: What facts about this case tell you that Neisseria meningitidis is one of the organisms that might be causing the infection? (Enter one factor per line.)
S: Age.
T: None of the factors that you mentioned are appropriate. The age of J. Smith is not evidence for the conclusion that Neisseria meningitidis is one of the organisms that might be causing the infection. After gaining more evidence about this case, the age of J. Smith will enable us to conclude that the organisms that might be causing the infection are E. coli (considering whether the infection was acquired while the patient was hospitalized), Pseudomonas aeruginosa, ...

(excerpted from Clancey [1987], p. 276)

Clancey's own commentary is clearer and more succinct:

T: The age of the patient will be relevant later, but it's premature to look at it now.

Although GUIDON contains explicit tutorial knowledge, the tutorial knowledge remains too close to the domain knowledge. Using the language of pragmatics, one could say that the problem with the text above is that human speakers do not say all of the information that they know. Using the language of text generation, we could say that an additional step is

necessary to convert the tutorial goals into language. Even though the text produced by GUIDON can be wordy and unnatural, it can still be useful to the experienced user, just as compiler error messages can be useful to programmers in spite of their stylized and repetitive nature.

1.1.6 Woolf: Procedural representation of tutorial knowledge in MENO-TUTOR

MENO-TUTOR [Woolf 1984, 1987] combines tutorial knowledge and discourse knowledge in a procedural representation. The action of tutoring is represented by several levels of augmented transition networks. The topmost ATN defines the basic teaching cycle for each topic: introduce a topic, teach it, identify and repair misconceptions, and complete the topic. The ATN formalism does not differentiate between sequential steps and alternatives. At run time, the student's input will determine whether the tutor will follow the teaching segment, the misconception segment, or possibly both. If more than one method is available for teaching a concept or there is more than way to express a method in text, each would follow a separate path through a lower-level ATN. Dialogue-related functions such as issuing acknowledgments and non-generation related functions such as parsing the student's input are interspersed with tutorial decisions.

Several methods of teaching are defined, including giving the student some general or specific knowledge, proposing an analogy, and suggesting an example. In the misconception segment, one of the creative tutorial methods is the "grain of truth correction." When the student's response is unexpected, MENO-TUTOR can respond cooperatively by jumping to a different part of the ATN.

MENO-TUTOR is related to CIRCSIM-Tutor in the way it handles unexpected inputs. Both systems can change or augment a plan when an unexpected input is received. Although MENO-TUTOR is hierarchically organized, its operation is essentially sequential; it

has no equivalent of intention-based planning. Additionally, MENO-TUTOR does not allow the representation of complex tutorial methods such as pointing out a contradiction to the student except through hard-coding a series of states. MENO-TUTOR is intended for domains where the tutorial methods are simple and content-independent. Although the use of content-independent methods was common in traditional CAI systems, they no longer seem adequate for representing the complex tutorial methods employed by human teachers.

Text produced by MENO-TUTOR is largely coherent because the ATNs are designed so that every path through them produces a reasonable response to the student's statements. The hierarchical organization makes it easier to validate this claim.

The original plans for MENO-TUTOR were never fully implemented. Input is done with menus or by giving the system hand-coded semantic forms. Text realization is done by string substitution.

1.1.7 Tutorial goals in plan-based systems

Recently a number of plan-based ITSs have been described in the literature. It is interesting to note that although these systems tutor students in different domains, they tend to have similar sets of primitives.

For example, Van Marcke [1990] lists the following ways to remediate a student error in the GTE ("Generic Tutoring Environment") system:

- Give correct answer
- Give explanation
- Give a hint
- Look at subproblems
- Diagnose error
- Retry without any of the above

This system contains four ways to clarify a concept:

- Clarify by description
- Clarify by example
- Clarify by simulation
- Clarify by analogy

Much of the content of this system is contained in the domain knowledge objects. For example, typical errors, generalizations, counterexamples and other types of tutorial information are stored in the system and described instead of having text generated about them on the fly.

Chevallier [1992] describes a system for tutoring elementary statistics using the following set of tutorial operators:[1]

- Question
- Acknowledgment
- Neutral commentary on the student's response
- Advice
- Hint
- Personalized explanation
- Canned explanation from syllabus
- Summary of proof or calculation

In Chapter 4, we will see that although these terms can be used to describe CIRCSIM-Tutor output, most of them are not CIRCSIM-Tutor primitives. Each of these systems is intended for teaching a topic which does not require as deep or extensive a domain model as CIRCSIM-Tutor. Each uses canned text or a similar template mechanism for output. CIRCSIM-Tutor differentiates the tutorial level, or method of teaching, from the discourse level, or type of interaction with the student. For example, the *show-contradiction* method can be implemented with questions, as in a Socratic dialogue, or in a single paragraph as a

[1] My translation.

monologic explanation. One instantiation of a tutorial method may be a multi-turn method while others are not. Additionally, CIRCSIM-Tutor combines output from consecutive primitives to form cohesive turns. Although the output of CIRCSIM-Tutor contains hints, explanations and similar categories, these phenomena are built from smaller primitives.

1.2 Text planning: intentional and decompositional approaches

1.2.1 Architecture of text generation systems

Text can be generated from a variety of internal representations. When the internal representation is isomorphic to the surface text or close to it, the term *template filling* is often used. A problem with template filling is that there is no level of linguistic abstraction, i.e. no way to represent the relationship between alternative ways of saying the same thing. Furthermore, the use of template filling makes it difficult to improve coherence within a turn because there is no intermediate semantic representation to manipulate. However, many natural language systems continue to use template filling because it is simple, fast and, at the sentence level, robust [Reiter 1995].

Many recent developers of natural language generation systems have chosen a similar division of functions between the phases of a text generation system [Reiter 1994, 1996]. Although no two systems divide their functions in exactly the same way or use exactly the same terminology, the following chart gives an overview of the major functions which must be accomplished.

- *Content planning or content determination*
  Choosing which concepts will be included in the text to be constructed. Content planning is sometimes accomplished as a side effect of another process such as an expert system. In other cases, content planning is accomplished in concert with the following phase.

- *Discourse planning, paragraph planning or sentence planning*
  Determining the outline of the text to be uttered. This might include choosing the sequence of concepts to be expressed, determining sentence boundaries, and choosing which concept will be expressed by the main clause and which by an adjunct. The output of this phase is an abstract semantic structure for each sentence.

- *Lexical insertion*
  Instantiating each object or action in the abstract semantic structure with a word or phrase in the desired language.

- *Text realization*
  Creating a surface string from the internal representation. Word order, morphology and agreement are some of the important phenomena which must be taken care of during the text realization phase.

The amount of processing required can be greatly reduced by such a pipeline architecture. This is true despite the fact that the phases cannot be precisely delimited. For example, whether a particular concept should be expressed as one sentence or two, or whether a particular concept should be expressed as a subordinate clause or a prepositional phrase, are decisions which may fall in paragraph planning or in text realization, depending on the structure of the planner and the reasons for the decision. Although Danlos [1987] and others have provided examples showing that certain lexical items require an integrated treatment of these two phases, this does not invalidate either the general principle or the practical necessity of reducing the branching factor.

The pipeline architecture is often described as separating high-level planning, or "what to say," from low-level sentence construction, or "how to say it." This two-way distinction is more useful in systems where content planning and paragraph planning are integrated into one phase which determines "what to say." In that case the interface between text planning and text realization is a data structure which represents, roughly, "what is to be said." This structure is often called a *discourse tree*. CIRCSIM-Tutor, on the

other hand, moves gradually from "what to say" to "how to say it" through the use of tutorial plans, which explain how to teach a particular concept. Thus the tutorial planner decides "what to say" as well as a certain degree of "how to say it." Section 5.3.1 describes the architecture of CIRCSIM-Tutor. The planning process is outlined in Section 5.1.1.

1.2.2 Using a global planner for discourse planning

A crucial issue in text generation is ensuring that the generated text is *coherent*, i.e. that the parts of the text fit together to express an idea. Just as a sentence is not just a sequence of words, a paragraph is not just a series of sentences. Although attempts have been made to generate paragraphs relying on surface models of coherence (e.g. [Mann & Moore 1981]), none have been successful in the long run. Similarly, the work of Halliday and Hasan [1976], which describes surface phenomena found in coherent text, does not provide a model which can be used for generation.

One way to ensure that a text conveys a single idea is to derive it from a high-level communicative goal, such as *to inform the hearer of X* or *to ensure that the hearer knows X*. The roots of this approach to text generation are found in speech act theory, a field of philosophy which considers speech as a type of act. This idea is usually attributed to Austin [1962]. Searle [1969] systematized Austin's work and has had more direct influence on the use of speech act theory in linguistics [Levinson 1983, p. 237]. In speech act theory, an utterance is an action by means of which a speaker does something to a hearer. In text generation the generation system is the speaker (or writer), and the user is the hearer (reader). In the case of a tutoring system, the speaker and hearer are often referred to as tutor and student.

Planning, or plan construction, is the process of finding a sequence of actions which

will transform a given state of the world into a desired state. Planning permits an agent to consider future actions before executing them. If we consider utterances as actions, then we can use planning algorithms to generate text. Other work on reasoning about actions also becomes applicable [Cohen, Perrault & Allen 1982]. Cohen & Perrault [1979] give a detailed derivation of sentences such as *Tell John to give me the key* from the goal of informing the hearer about something.

Another early work using planning to generate text is that of Appelt [1985]. Appelt's goal was to provide a formal proof that the sentences he generated indeed fulfilled their discourse goals. A lot of his attention went to generating correctly referring noun phrases. He generates one or at most two sentences for each discourse goal. Appelt uses a hierarchical planner based on the one devised by Sacerdoti [1977] to derive an abstract representation from a set of basic axioms. He uses a functional unification grammar to generate the final text from the abstract representation.

The earliest formal planning system is the STRIPS planner of Fikes and Nilsson [1971]. In the text planning domain, a STRIPS-style or "classical" planner begins with the goal of generating text to implement a given speech act, such as *to inform X of Y*. The planner has operators which can replace this goal with smaller subgoals when appropriate preconditions are satisfied. Bottom-level subgoals add to an intermediate representation such as the discourse tree defined above. When a set of subgoals equivalent to the original goal have been simultaneously satisfied, the discourse tree is input to the realization phase, which converts the intermediate representation to natural language text. [Hovy et al. 1992] The realization phase can use any computer-compatible grammatical theory, and may or may not use a planning mechanism. Two popular choices for the underlying grammar are functional unification grammar and Halliday's systemic grammar.

Using planning as a model for text generation gives rise to the problem of combinatorial explosion. One way to reduce the number of choices needed to generate a text is to obtain a general plan at a high level of abstraction and then refine each step independently. This method, known as *hierarchical planning*, was first implemented by Sacerdoti [1974] in the ABSTRIPS system. A potential problem with this method is that contradictions between the refinement of one step and the refinement of another can cause significant backtracking. This problem was somewhat ameliorated in the NOAH system of [Sacerdoti 1977]. NOAH had *constructive critics* which attempted to fix contradictory refinements to avoid backtracking. Depending on how the parts are implemented, the common arrangement of two systems connected by a discourse tree may or may not be correctly considered a hierarchical planner.

An improvement in efficiency can be obtained through the use of *partial-order planning* [Weld 1994]. Instead of representing a plan as a sequence of steps, a partial-order planner uses a graph representing only necessary relations between the plan steps. Before executing a partial-order plan, one chooses a specific linearization. The goal of partial-order planning is to reduce backtracking by avoiding unnecessary commitments to the order in which subgoals are satisfied. Partial-order planning has the potential to make discourse planning more efficient because this issue occurs often in discourse planning. For example, if a phrase or clause has more than one prepositional phrase attached, the order in which text is chosen to realize the adjuncts is often arbitrary.

UCPOP (pronounced *you-see-pop*) [Barrett et al. 1994] is a partial order planner which permits universal quantification in the preconditions, effects and goals of plan operators. It also permits conditional effects. These features make it easier to represent actions in the everyday world. Penberthy and Weld [1992] prove that UCPOP is sound and

complete, two attributes which simplify implementation by eliminating the possibility of generating a plan which doesn't work or missing a correct plan. DPOCL is a system derived from UCPOP and intended specifically for text generation [Young & Moore 1994b; Young, Moore & Pollack 1994; Young, Pollack & Moore 1994]. A recent addition is to this line of research is LONGBOW, a new planner based on DPOCL and intended specifically for discourse planning [Young 1994]. LONGBOW permits nested functions in plan operators, a feature essential for text generation but which is not implemented in UCPOP. LONGBOW provides the ability to use a domain-independent declarative representation for discourse plans instead of writing a custom discourse planner for each application [Young & Moore 1994a].

1.2.3 Two approaches to planning

The planner formalism can be applied to text generation in many ways. In *decompositional* planning, the prerequisites of a plan operator specify a speech act which the speaker wishes to perform, and the body of the operator gives a set of subgoals, which, executed in sequence, will result in the successful completion of that act. In *intentional* planning, the prerequisites specify a change in the hearer's mental state which the operator should achieve if satisfactorily executed. Decompositional and intentional operators can coexist in the same system. For example, "get the tutor to teach X" and "get the student to know X" are decompositional and intentional ways of looking at the same problem. Precompiled decompositions of a problem are often known as *schemata*. A schema consists of a list of semantic primitives or other schemata which can be used to realize a communicative goal. Schemata are useful whenever one can identify a sequence of smaller parts from which one can build a text.

If schemata are not used, a different method must be used to structure the text. For

example, *discourse structure relations* (DSRs) can be used wherever one can identify a set of rules relating the parts of a text to one another. A discourse structure relation describes the relation between two pieces of text, such as *example-of* or *elaboration-of*.

The syntax of the CIRCSIM-Tutor plan operators is described in Section 5.3.8. Section 5.1.2 describes the motivation for the use of schemata in CIRCSIM-Tutor. Note that the implementation of schemata has changed over time. In particular, the schemata used in CIRCSIM-Tutor are more flexible and powerful than the schemata in many earlier systems.

In the following sections we look at some important early text planning systems using both paradigms.

1.2.4 McKeown: Schemata

The use of schemata in generation is usually attributed to McKeown [1985]. McKeown's system generated text for three discourse goals, *to define*, *to describe* and *to compare*. The generated text followed one of four schemata, *identification*, *constituency*, *attribution* and *comparison*. A set of rules was used to determine which schema was used for each discourse goal. For example, depending on the object *X*, the goal *define X* could be expanded using the constituency schema or the identification schema. The goal *describe X* could use the constituency schema or the attribution schema.

Each schema consisted of a list of lower-level *predicates* or semantic primitives. Seven predicates are implemented: identification, attribution, constituency, analogy, illustration, evidence, and amplification. (Predicates and schemata of the same name are distinct.) Indefinite repetitions of a predicate were permitted. For example, the identification schema consisted of the following list:

Identification:
(Analogy | Constituency | Attribution)*
(Illustration | Evidence)+
Amplification | Analogy | Attribution
Illustration | Evidence

Recursion was not permitted except in the comparison schema, which permitted exactly one level of recursion, namely using one of the other schemata to describe each of the objects to be compared.

When more than one proposition in the data base would fit in a given slot in the schema, Sidner's [1983] theory of immediate focus was used to pick the one which was most likely to lead to coherent text. A functional unification grammar was used to translate the instantiated schema into grammatical text.

1.2.5 Paris: Multiple strategies for describing a concept

Instead of always using the same schema to answer a given question, Paris [1985] noticed that writers use more than one strategy to describe an object depending on the user's level of expertise. She identified two of these strategies, the descriptive strategy, preferred by experts, and the procedural trace, preferred by novices. Instead of assuming that each user fell completely in one of these categories, she allowed the user to be knowledgeable or not about each subpart of the object being displayed. Thus her system generated a mix of descriptive and procedural trace information customized for each user.

Paris [1988, 1993] extends McKeown's planner in three ways. First, she has an explicit user model. The user is classified on a continuum from novice to expert, and a list is also kept of which concepts the user is presumed to understand. Second, Paris' system does not use pure schemata. Under some conditions, it traces links in the data base and produces a text structured according to the kind of links it finds as opposed to a

predetermined schema. This strategy is used to describe how something works [Paris & McKeown 1987]. Third, Paris' system has full recursion.

1.2.6 Moore: Intention-based planning

Moore's system is described by Moore and Paris [1989] and Moore and Swartout [1991]. This system is used as the text generator for the expert system shell EES (Explainable Expert System) [Neches, Swartout & Moore 1985; Paris 1991]. As the input is processed, Moore builds a history tree containing four kinds of information: intermediate goals (intentional structure), information about dialogue focus (attentional structure), rhetorical structure, and user model information. The attentional information which must be saved is defined in terms of the focus theory of Grosz and Sidner [1986]. As the system plans explanations to achieve discourse goals, it saves a trace including the reasons for its decisions. Then, when the user asks a follow-up question, it can respond based not only on what it believes the user already knows, but on how this particular conversation is shaping up. For example, it can use previous failed attempts to explain something to get a better idea of what the user really wants to know and how it could be explained.

1.2.7 Hovy: Using discourse structure relations for generation

Hovy [1988, 1991] wanted to generate connected discourse using the concept of axioms from Appelt's work instead of schemata. To obtain the axioms he needed to connect larger units of text, Hovy drew from the Rhetorical Structure Theory (RST) of Mann and Thompson [1986, 1988]. Mann and Thompson had looked at the relations between clauses of a sentence as well as the relationship between sentences in connected discourse. They labeled a large number of these relations, such as CIRCUMSTANCE,

ANTITHESIS and SEQUENCE. For example, the sentence *they laughed when I sat down to play* uses the CIRCUMSTANCE relation to connect the satellite *I sat down to play* to the nucleus *they laughed*. Although RST is an open system which can be augmented as needed based on the context in which the theory is being applied, the basic set of relations is fairly constant.

One major network of DSRs was developed by a cross-disciplinary group of researchers and reorganized by Maier and Hovy [1991]. The major influences are Mann and Thompson's RST taxonomy and Martin's taxonomy of English conjunctive relations [Martin 1983]. Following Halliday's ideas, there are three major categories of DSRs: *ideational*, *interpersonal*, and *textual*. Ideational relations, which include various subtypes of ELABORATION, CIRCUMSTANCE and SEQUENCE, relate two concepts. Interpersonal relations, which include categories such as INTERPRETATION and ENABLEMENT, describe the speaker's relation to the proposition. Textual relations relate two pieces of discourse. Many function words, such as *likewise*, *rather* and *but*, are instantiations of textual relations. Some relations can be used to advance more than one theme depending on the context. For example, CONCESSION ("although") can refer to a fact (ideational) or to another part of the text (textual).

In order to avoid the unconstrained branching factor which would arise if the system could choose any of Mann and Thompson's relations each time the discourse tree was expanded, Hovy restricts the branching factor by assigning to each RST relation a small number of other relations, called *growth points*, that the system can choose for its next expansion possibility. Hovy points out that this reduces the full generality of his system and makes generation via RST look more like schemata with full recursion. The PENMAN text realization system [Mann 1985] is used to generate English text from the preferred

discourse tree.

In order to get back to the full generality of the RST relations, another way must be found to reduce the branching factor. Hovy and McCoy [1989] use focus rules derived from Grosz and Sidner's focus theory [McCoy & Cheng 1991] to reduce the number of possible RST relations and topics to instantiate them with at each choice point.

Recently, many researchers, beginning with Moore and Pollack [1993], have pointed out problems with using RST for text generation and suggested various alternatives.

1.2.8 Rösner & Stede: Combining schemata and DSRs

There is no *a priori* reason why DSRs and schemata cannot be used in the same system. This approach is used by Rösner and Stede [1992] in their research on the generation of automobile maintenance manuals. They represent each maintenance procedure as a schema with three subdivisions:

- Preconditions
- Plan steps
- Postconditions

Two types of optional annotations can be used to amplify each schema step:

- Warnings of various degrees of severity
- Hints about potential errors or failures

Each schema step is then broken down further using RST relations.

1.3 From plan to text

This section provides background information for understanding how semantic forms are turned into text in CIRCSIM-Tutor.

1.3.1 Turn planning as an instance of paragraph planning

CIRCSIM-Tutor uses a schema-based planner for tutorial planning. Once the semantic forms for a turn have been chosen, they are organized into turns by a second process, the turn planner. Thus, from the turn planner's point of view, the tutorial planner has accomplished the content generation step.

In theory, any paragraph planner could fulfill the needs of the turn planner. In practice, a considerably simpler system is sufficient. Although it is exciting to think about complex ways to combine semantic forms within a turn, we do not currently have tutorial planning operators or a lexical insertion module sophisticated enough to use such a feature.

Since most of the sentences generated by CIRCSIM-Tutor represent discrete events, such as the change in value of a variable, we can simplify the processing by generating one sentence per semantic form in many cases. A small number of rules is provided for cases where significantly better text can be generated by combining semantic forms. The output of the turn planner uses a basic schema for a turn derived using the methodology of Conversation Analysis (Section 1.3.4). The turn planner is responsible for calling the lexical insertion and text realization phases.

1.3.2 Levels of semantic forms

Many researchers have independently discovered the value of having an intermediate semantic representation which is both domain-independent and independent of any specific human language. Robin [1994] follows the suggestion made by Elhadad [1992] that the use of a declarative formalism for representing the transformations between levels would lead to a cleaner implementation. In Robin's system, the *deep semantic form* describes what happened. It is language-independent and uses concepts from the domain knowledge

base and concepts derivable from them.

The *surface semantic form* describes what we want to say about what happened. The surface semantic form is intended to be domain-independent, as basic ontological relationships such as before/after, causality, and so forth can be used in any domain. The surface semantic forms are described using the representation described by Fawcett [1987], extending the systemic grammar of the verb presented by Halliday [1985]. The surface semantic form is independent of any specific human language. It is obtained from the deep semantic form through the use of a set of functional unification rules.

The *deep syntactic form* uses a representation which is syntactically identical to the surface semantic form, but which contains fields representing syntactic roles in addition to fields representing types of objects and events. To obtain the deep syntactic form, Robin uses another set of functional unification rules to choose lexical items for the actions in the surface semantic forms. Choosing a verb determines the argument structure, so that the sentence is more nearly determined.

Finally, the *surface syntactic form* is the familiar syntax tree obtained from the text realization process. For this purpose Robin uses the FUF system of Elhadad [1992, 1993].

Although Meteer [1992] uses a very different method for text realization, she also posits a level of semantic structure which she calls Text Structure. The Text Structure shows the relationship between the events involved in a sentence and the relationships between those events and the objects they involve.

Plan operators in CIRCSIM-Tutor correspond to surface semantic forms in Robin's terminology. Since the fundamental knowledge representation in CIRCSIM-Tutor is a schema indicating how to teach something, it is already a discourse-based representation. We use a lexical insertion process similar to Robin's to choose verbs and case frames for

sentences. The output of the lexical insertion process, a set of deep syntactic forms, or surface semantic forms with verbs and their arguments filled in, is then input to FUF for text realization.

1.3.3 Systemic grammar as an approach to text realization

The work of Halliday [1985] and other proponents of systemic grammar is useful in text generation for two reasons. First, as a functional linguist, Halliday is interested in the use of language in context and the relationship between syntactic structures and the functions they convey. In addition, the British school of systemic linguistics, of which Halliday is a member, cares about whether their theories are implementable and whether they match reality.

A systemic grammar is a constraint-satisfaction system for describing a language, implemented as a series of (possibly recursive) AND-OR graphs. Each node or *system* (hence the name), contains a piece of procedural code called a *chooser*. The chooser looks at all the available data. Then it sets the values of one or more *features* which will determine which of the possible successor nodes will be accessed next. For example, one of the choosers in the verb network determines whether the verb to be generated should be in the indicative or the imperative mode. If the indicative mode is chosen, the next chooser decides whether the verb will be declarative or interrogative. Recursion is needed, for example, so that a prepositional phrase can contain another prepositional phrase as a modifier. When a piece of output text has been fully specified, it is produced by a realization routine called by the node which determined this fact. Systemic grammars can model complex syntactic phenomena without the use of transformations.

The FUF text realization package [Elhadad 1992, 1993] uses a systemic grammar of English called SURGE. FUF uses functional unification as its underlying formalism.

1.3.4 Sinclair & Coulthard: Conversation Analysis

Conversation Analysis, a sociological approach to dialogue, is an invaluable theoretical resource for dialogue planning. Conversation Analysis is more concerned with describing how turns accrete to form a conversation rather than the hierarchical structure of the conversation as a whole. A *transaction* corresponds to the discussion of one topic. In our system, the correction of each variable is a topic. An *exchange* is the smallest unit of dialogue, and *moves* are the smallest independent goal units. Note that turns are not coterminous with exchanges; in fact, a turn often ends one exchange and starts another.

Like RST, Conversation Analysis is a methodology rather than a theory with a specific content. The work of Sinclair & Coulthard [1975], to which I was introduced by Cawsey [1992], is particularly relevant to CIRCSIM-Tutor because it contains an analysis of the speech of teachers. Stenström [1994] provides an alternate set of definitions for the levels posited by Sinclair and Coulthard. Traum and Hinkelman [1992] use Conversation Analysis as a guide to parsing. Although the problem they are studying is inverse to ours, their use of Conversation Analysis is similar.

In CIRCSIM-Tutor, the Conversation Analysis methodology has been used to develop a schema for the structure of a turn. The turn planner builds turns which fit this schema.

1.4 Planning and replanning in an ITS

1.4.1 Using global planning with replanning for dialogue generation

One can think of the basic planning cycle of an ITS as driven by the student's responses or by the tutor's goals. Wenger [1987, section 18.2] refers to the two possibilities as opportunistic and plan-based pedagogical styles. Since even a plan-based ITS must take the student's responses into account, the true difference between the two

styles is the fact that in a plan-based ITS, the tutor's actions are derived from a global plan. In a text-based ITS, the use of such a global plan can create a high-level organization for the text and thus a sense of high-level coherence. The plan-based style is also a natural one to use for an ITS such as CIRCSIM-Tutor where tutoring must proceed in a specified sequence.

Yet many researchers [Bilange 1991; Gerlach & Horacek 1989; Jönsson 1991; Jullien & Marty 1989] have shown that planning dialogues is different from planning monologues. The crucial issue is that dialogues cannot be completely planned in advance because the tutor cannot predict how the student is going to respond, making it impossible to know how the conversation will evolve.

This fundamental point has echoes in three aspects of planner design: the need to interleave planning and realization, the need to maintain global planning strategies and multi-turn plans regardless of the student's input, and the need to maintain a coherent conversation. The ability to interleave planning and realization is important because it is wasteful to plan future conversational turns in detail when the conversation may take a different tack before those turns are reached. The use of global strategies and multi-turn plans is a significant factor in achieving the tutor's pedagogical goals as well as contributing to coherence. Finally, in order to maintain a helpful and coherent conversation, the tutor must occasionally make responding to the student a higher priority than continuing with the plan.

1.4.2 Wilkins: Replanning in SIPE

Wilkins' SIPE system [1988] is an advanced classical planning system used for activities such as robot planning, not a text planning system. Wilkins' work is relevant to ours because it is one of the few planning systems using the classical model which can

cope with unexpected events happening during plan execution. For example, the robot might arrive at a room and find the door locked. Wilkins' system copes with unexpected events by adding what he calls "Mother Nature nodes" to the plan in progress. Wilkins presents a taxonomy of actions which the plan execution module can attempt in order to correct the plan, such as grafting new steps onto a plan in progress.

The replanning methods in CIRCSIM-Tutor are conceptually similar to those SIPE methods which apply to text planning. The fact that SIPE can cope with unexpected input and continue goal-oriented processing is very similar to what CIRCSIM-Tutor needs to do. On the other hand, one aspect of robot planning which is different from text planning is that the tutor cannot "unsay" anything. If the robot drops a screw, it can pick up another screw and try again with few or no untoward consequences. However, if the student does not understand a question and the tutor decides to ask it again, it is likely that the second attempt will need to be somewhat different from the first in order to be cooperative in the sense of Grice [1975]. In addition to choosing a different way to ask the question, our human tutors sometimes use formulas such as "let me try again" to indicate that they know the question is a repetition.

1.4.3 McCalla: Replanning in a simple domain

Peachey and McCalla [1986] give an algorithm for updating a plan based on input from the student. This algorithm is similar to the one described in Chapter 5 for CIRCSIM-Tutor. The student model consists of a set of atoms representing concepts the student needs to learn and misconceptions the student possesses. The system teaches by displaying canned text and asking a question. Peachey and McCalla update the plan based on the student's responses. Spontaneous learning by the student causes the planner to drop a subgoal which is no longer necessary. A wrong answer can cause a remedial operator to

be added to the plan to remediate a misconception.

Huang and McCalla [1992] add a truth maintenance system to the design proposed by Peachey and McCalla. The student model is stored as a set of clauses. After each student input, the truth maintenance system is used to update the student model and determine how the instructional plan should be updated.

As in many fields, logically complete algorithms which work for a small, uncomplicated knowledge base are not necessarily practical in real systems. Nevertheless, these papers are worthwhile because they point at a principled way of revising plans. The replanning algorithm in CIRCSIM-Tutor is consistent with the one described by Peachey and McCalla.

1.4.4 Jullien & Marty: Replanning in a dialogue system

Jullien and Marty [1989] describe a method for plan revision in dialogue similar to that used by CIRCSIM-Tutor. Jullien and Marty describe an advice-giving system which asks the user for financial information as it tries to draw up an investment plan. They maintain three data structures: a tree showing the current plan; an attentional stack, listing the potential foci when the question is asked; and an expectation stack, listing all of the communicative goals the user could be executing at each level of the tree. Each potential unexpected answer, such as asking for a definition or an explanation instead of answering the question, is connected to a tree transformation. When a specified type of deviation occurs, the corresponding transformation is executed to create a new plan tree. In the example above, the transformation would add a communicative goal to the tree to provide the desired definition or explanation before returning to the previous topic. Jullien and Marty state that the expectation stack is used to validate potential responses and the attentional stack is used to reduce the possible interpretations of the user's statements.

In CIRCSIM-Tutor, each potential deviation points to either an adjunct schema which will be added to the top of the stack or to a replacement schema. In the example above, the adjunct schema would contain the semantic forms necessary to provide the definition or explanation.

In contrast to Jullien and Marty, Gerlach & Horacek [1989] and Chu-Carroll and Carberry [1995] use an intention-oriented method for determining the best response to make in a dialogue. Smith [1992] uses a method directly based on theorem proving, but which only works in a very simple domain.

1.5 Plan-based explanation systems

Our work is different from other large-scale generation systems [Moore 1995; Cawsey 1992; Maybury 1992] in that the latter are explanation systems which accept follow-up questions rather than tutoring systems. Thus they do not need to generate the variety of teaching mechanisms which CIRCSIM-Tutor requires. Furthermore, in each of these systems, the dialogue terminates when the student is satisfied. In CIRCSIM-Tutor, the student must prove through correct answers to questions that the material has been mastered.

Conversations with CIRCSIM-Tutor are generally longer than conversations with an explanation system. In the systems mentioned above, a dialogue usually includes an explanation by the system followed by a small number of follow-up questions. To complete one CIRCSIM-Tutor problem, the student must eventually give correct values for seven variables at each stage. It is rare for a student to be able to do this without additional dialogue. Because of the greater length of the conversation and the fact that the student is required to demonstrate active understanding, the issue of updating the user model during a conversation, which these systems do not do, becomes more important.

1.5.1 Cawsey: EDGE, a dialogue-based explanation system

In many ways, Cawsey's EDGE system is the closest in spirit to the work described here. It teaches concepts from a causal model and embeds the content in a naturally structured conversation. However, since CIRCSIM-Tutor needs to cover more complex content in each tutoring session and has a greater variety of teaching methods to do it with, there are some structural differences.

In Cawsey's system, the Conversation Analysis paradigm is used as the main structuring principle for the dialogue, with the content material subordinate to it. Every generated turn is connected to the explanatory goal which generated it. One never needs to backtrack from an explanation subgoal, so the necessary explanatory goal is always available. When the student gives a wrong answer and the tutor corrects it, the correction turns are nested under the same explanatory goal. This structure is not useful in CIRCSIM-Tutor, where most of the text consists of the tutor correcting the student's misconceptions. We do not want to package the correction turns with the original turn because there may be a change of topic involved, for example, when the tutor identifies a misconception and tries to correct it.

Additionally, CIRCSIM-Tutor needs to maintain a deeper tree of content goals than EDGE in order to cover its material with appropriate depth and variety. Thus it is more sensible for us to consider the tutor's goals as primary. We use material from Conversation Analysis in a "coroutine" fashion to maintain natural conversation patterns between the tutor and student as the tutor's goals are satisfied.

We share Cawsey's use of the Conversation Analysis methodology [Sinclair & Coulthard 1975; Stenström 1994] as a guide to conversation planning.

1.5.2 Maybury: The TEXPLAN system

Maybury's TEXPLAN system [1991, 1992] illustrates several ways of explaining a concept, such as the use of various types of definitions and descriptions. One aspect of TEXPLAN which resembles our system is that plan operators may match on either the *header* field, which identifies the speaker's communicative act, or the *effects* field, which contains the intended effect on the hearer. From the point of view of CIRCSIM-Tutor, this means that a plan operator can be chosen either because it will express the tutor's communicative goal or because it will have a desired effect on the student's understanding. These two aspects are useful in different circumstances.

Although explanation is one of the methods which CIRCSIM-Tutor uses to enhance the student's understanding, explanations *per se* are not a major focus of our work. We are more interested in generating a relevant explanation at an appropriate point in the conversation. In general, CIRCSIM-Tutor only gives an explanation after the failure of a method such as hinting, which requires more active participation on the part of the student.

1.5.3 Moore: Using intention-based planning to generate follow-up questions

The goal of Moore's PEA system [1995] is to be able to handle follow-up questions more intelligently by understanding the meaning of the student's response in context. Thus Moore needs to know which part of the tutor's statement the student understood in order to know what to say next. Although Moore's basic design has much in common with CIRCSIM-Tutor, this aspect of it is not relevant to us.

Moore uses plan operators described according to their intended effect on the hearer. Moore sets up her prerequisites the way she does specifically for the use she will make of them in generating follow-up questions. But we generate follow-up questions the

same way we generate any other text, i.e. from semantic forms. Moore does not try to discern the accuracy of these beliefs or update her student model based on them. She assumes that the desired alteration in the student's belief always happens, and she does not worry about updating the student model. These simplifications work for her because she is not generating her follow-up questions from scratch. In exchange for these restrictions, she can implement a sophisticated system for generating follow-up questions. CIRCSIM-Tutor needs to be able to generate many types of discourse mechanisms, not just explanations and follow-up questions.

1.6 Causal and functional models

There is no such thing as a perfect model. A model is only useful for a given purpose. Sometimes multiple models are required for different purposes. Sometimes when one tries to create a deeper model, the nature of the model changes. What appears to be the most important factor in choosing a model is to make sure that the system, and thus the student, is using the model an expert would use to solve the same problem. A causal model can contain two kinds of knowledge, knowledge about the causal sequence of events and knowledge relating factors inside a given event. As we have seen in Section 1.1.4, the latter is sometimes called functional knowledge. In this section, we examine some systems where authors have found it necessary to make this distinction.

1.6.1 Brown, Burton and Zdybel: Causal and functional modeling in a meteorology tutor

The meteorology tutor of Brown, Burton and Zdybel [1973] attempts to represent causal and functional knowledge in order to answer questions involving causal knowledge. The authors state that meteorology is a good subject area for studying causal reasoning because major processes such as evaporation, condensation and rainfall can be described

at an elementary level using qualitative reasoning. Causal and functional knowledge is represented by an augmented finite-state automaton for each process, with state transitions representing the events. For example, the ATN for humidity of saturation contains three states: *decreasing*, *stable* and *increasing*. Since humidity of saturation is determined by air temperature, the link from *stable* to *decreasing*, for example, is labeled "air temperature decreases." The transition conditions can also refer to global predicates which are used to represent the current discourse context.

The complete model includes about twenty interrelated ATNs and can calculate values for variables such as rate of evaporation, absolute humidity, rate of condensation and relative humidity. In addition, one of the ATNs can cycle through the steps of the coalescence process, which causes rainfall.

After the student chooses the input conditions, the tutor runs the model. To permit the generation of complex text, the system constructs a tree containing all possible paths through the ATN, along with their associated text, before generating any text. This tree can be used to answer any question about the causal relationships between the events modeled by the ATN. In its current form, the system generates text by tracing the tree, adding discourse particles according to a set of rules, but the authors state that one could write a natural-language back-end to generate more natural text from the tree. The following is an example of the generated text.

(6) S: What happens when the water temperature drops to 22 degrees?
T: The air temperature decreases from 25 to 22 degrees because water temperature decreases. The humidity of saturation decreases from 24 to 20 mm Hg because air temperature decreases. The relative humidity increases from 100 to 120% because humidity of saturation decreases ...

(excerpted from Brown, Burton and Zdybel [1973], Appendix II)

To simplify the domain model, at each step transitions representing instantaneous changes

are processed before transitions representing an advance in time. The authors assert that distinguishing between transitions due to functional relations and transitions which advance the clock will keep the ATNs from looping. This distinction is still a key feature of today's qualitative physics. Although CIRCSIM-Tutor contains a causal model, it does not distinguish between these two types of transitions. Making this distinction would simplify the deeper levels of the concept map and permit the generation of better text for deep explanations.

1.6.2 Stevens, Collins and Goldin: Classifying students' misconceptions in meteorology

Stevens, Collins and Goldin [1979] started by modeling physical processes as scripts, which they describe as partially ordered sequence of events connected with temporal or causal connectors. For example, each step in the processes causing rainfall was described, and the steps were connected with relations like *precedes*, *causes*, and *enables*. The following example shows the high-level script for heavy rainfall.

> A warm air mass over a warm body of water absorbs a lot of moisture from the body of water.
>
> *precedes*
>
> Winds carry the warm moist air mass from over the body of water to over the land mass.
>
> *precedes*
>
> The moist air mass from over the body of water cools over the land area.
>
> *causes*
>
> The moisture in the air mass from over the body of water precipitates over the land area.
>
> (excerpted from Stevens, Collins and Goldin [1979], fig. 1)

The model is hierarchical, so that a step at one level can break down into a sequence or graph of steps at a more detailed level. Although this model could be used to query students about the steps leading to rainfall and teach the missing steps, it could not respond to students' underlying misconceptions.

The authors conclude that many of the students' underlying misconceptions could only be expressed in terms of a *functional* viewpoint. They decided that in additional to the linear view, it was necessary to describe the relationship between the variables involved in each major phenomenon. They used the following representation for the functional viewpoint. Each phenomenon has a set of *actors*. Each actor has a *role*. Each actor has a set of *attributes*. Some of the attributes are *factors* in the process being described. The *result* is always a change in the value of some factor. Finally, the functional viewpoint includes a description of the *functional relationship* between the factors and the result. For example, consider the evaporation phenomenon. The actors are a *moisture source*, which must be a large body of water, and a *destination*, which is an air mass. The factors are the temperature of the two actors and the distance between them. When the temperature of the moisture source goes up, the humidity of the destination air mass increases.

Stevens, Collins and Goldin used the functional viewpoint to analyze dialogues with live tutors. They discovered that the high-level structure of the dialogues is controlled by the scriptal viewpoint. But in order to correct student misconceptions within a step, tutors tend to shift to a functional viewpoint in order to ensure that the student understands the main actors, the important attributes and the relationships between them.

Finally, Stevens, Collins and Goldin attempted to identify misconceptions at the functional level which were shared by many students, including sixteen bugs just on the

subject of evaporation. For example, many students believe in the "cooling by contact" bug, i.e. that warm air touching cold land causes condensation. According to the model of Stevens, Collins and Goldin, since the remediation of misconceptions is accomplished using a functional viewpoint, we need to develop a functional model before we can isolate a set of conceptual bugs for CIRCSIM-Tutor.

1.6.3 STEAMER: Modeling via graphical simulation

STEAMER is an instructional tool for training people to operate the steam propulsion systems on large ships. Although the goal of the training is to teach the students to perform the necessary procedures correctly in both normal and problem conditions, this can only be accomplished by giving students a mental model of the steam plant and teaching them the relevant engineering principles. The propulsion plant occupies approximately one-third of the space on a ship. It is impossible for students to memorize several hundred procedures, and there is no way to pre-plan a procedure for every possible fault that could happen [Williams, Hollan, & Stevens 1981]. This is similar to the situation in CIRCSIM-Tutor: one cannot predict all of the failure modes of the human body, but we want the student to react correctly to any which arise.

STEAMER is an interactive graphics-based simulation. The goal of STEAMER is *conceptual fidelity*, trying to match the models used by experts, rather than physical fidelity [Hollan, Hutchins & Weitzman 1984]. According to the description given by Stevens and Roberts [1983], functional information relating the components of the model is built into the graphical simulation. Causal information is represented through the use of goal-oriented procedures, each of which is represented as a series of steps. Students can change not only parameters which they could affect in the real world but also derived values which cannot be directly influenced in the real world. Thus in addition to directly

simulating potential actions, students can use the simulation to increase their general understanding of the propulsion plant.

Although no details on the tutoring algorithm are given, the authors state that when the student makes a mistake, the relevant engineering principles are activated and the system generates a message such as the following:

(7) T: According to the principle which requires that whenever you admit steam into a closed chamber you should first align the drains, before opening value 13 you should align the drain valves FWD-E254 and FWD-E239.
([Stevens & Roberts 1983, p. 19])

As in GUIDON and the meteorology tutor, the text in STEAMER has been generated from a form close to the original domain knowledge.

Chapter 2
Introduction to the Baroreceptor Reflex Domain

The heart has its reasons, which reason cannot know. —*Blaise Pascal*

This chapter starts with some basic facts about cardiovascular physiology. Then we describe the baroreceptor reflex problems which constitute the domain of CIRCSIM-Tutor. To give an idea of the range of the system, we enumerate all of the problems of the simplest type and give some representative examples of more complex problems. We give rules for solving the problems and give a detailed solution trace for one problem. (Appendix A contains detailed solutions for all of the simple problems and a number of others.) We point out the influence of the model on the type of language which CIRCSIM-Tutor can generate.

2.1 Teaching cardiovascular physiology

One of many specific topics which medical students must learn is how blood pressure is regulated in the human body. Human life is only compatible with a specific range of blood pressures. When something happens to change the blood pressure, such as increasing or decreasing the volume of blood in the body, the body tries to compensate. The negative feedback loop which controls this process, known as the *baroreceptor reflex*, is a difficult topic for students in the first-year physiology course.

For many years, Professors Joel Michael and Allen Rovick of Rush Medical College have been experimenting with a variety of methods to help students understand this process, including reading, lectures, and various types of problem-solving exercises. For

the last ten years, they have also included regularly scheduled computer laboratory sessions. During these sessions, they assist students in using CAI programs to solve problems about the baroreceptor reflex similar to those which will appear on course examinations. Several years ago, they became interested in using artificial intelligence techniques to allow a program to conduct a conversation with the student instead of using canned text. This was the genesis of the CIRCSIM-Tutor project.

CIRCSIM-Tutor uses a simplified model of blood circulation to help students learn the basic concepts of blood pressure regulation. As we have seen in the previous chapter, what is important for tutoring purposes is not a perfect model of the domain but a model of the domain that matches what experts use. Not only can any model be further refined, but the type of information in the model changes as the model becomes more detailed. Although higher-level models of the baroreceptor reflex deal with the values of variables, lower-level models deal with intracellular events. Which of these models is most appropriate at a given moment depends on the question being answered or the problem being solved. Our domain experts teach students using the highest-level model possible. They drop to a lower level only when the student is having difficulty, and they can always explain the baroreceptor reflex without resorting to the intracellular level. In addition, no model is a perfect mirror of reality. For example, CIRCSIM-Tutor v. 2 does not consider the influence of mean arterial pressure (MAP) on stroke volume (SV). In version 3, this influence is taken into account, providing a more accurate but also a more complex model.

2.2 A layperson's guide to the baroreceptor reflex

2.2.1 Defining the baroreceptor reflex

The heart is an intermittent pump which repeatedly moves a small volume of blood forward in one fixed direction in a circular path through a closed system of blood vessels.

The heart pumps the entire blood volume at a steady rate during a half second and then pumps nothing, with no pressure in the system, during the next half second. During the pumping phase, or *systole*, the heart contracts and empties; during the quiet phase, or *diastole*, it relaxes and is refilled. It takes approximately one minute for the blood to circulate through the body. Externally caused events, such as administration of a drug, change blood pressure by changing the need for blood in different parts of the body. Since human life is only compatible with a small range of blood pressures, the body immediately tries to return the situation to normal.

A mechanism which causes a parameter to return toward a preset value after a disturbance is called a *negative feedback system*, "negative" because a portion of the difference between the original value and the new value is applied as a correction to moderate the new value. Every negative feedback system attempts to maintain the value of a *regulated variable* by changing the value of other variables known as *controlled variables*. The negative feedback system which regulates blood pressure is called the *baroreceptor reflex*; the regulated variable is the mean arterial pressure (MAP). A *reflex* is a hard-wired, stereotyped response to a well-defined stimulus. The prefix 'baro' means "pressure," as in "barometer." The baroreceptor reflex is an *autonomic* reflex because it is mediated by the autonomic nervous system. It is a *homeostatic* mechanism because it attempts to return MAP to its initial value.

To teach the student about the baroreceptor reflex, CIRCSIM-Tutor concentrates on the causal relationships between the regulated variable, the controlled variables, and a small number of intermediate variables. The value of MAP is measured via anatomical structures in the neck known as *baroreceptors*. A change in the pressure sensed by the baroreceptors changes the signal sent through the nervous system to the *effector organs*,

which changes the values of the parameters measured at these organs. These parameters are the controlled variables.

2.2.2 Basic anatomical structures and physiological parameters

Blood carries oxygen from the lungs to the body tissues. The pumping action of the heart carries blood from the lungs to the capillaries in the body tissues, and back again. The following description introduces some of the anatomical objects which are important to CIRCSIM-Tutor.

Blood is pushed out of the heart from the *left ventricle*. From the left ventricle it enters the *arteries*. From the arteries the blood flows through the *arterioles*, which are smaller in diameter, and thence to the capillaries, which are even smaller. From the capillaries the blood gives up its oxygen to the cells in the body tissues.

After giving up its oxygen, the blood returns to the heart via increasingly large veins in the venous system. The large *central veins* (also called the *great veins*) serve as a reservoir for blood. The central veins are *compliant*. When blood volume increases, they expand like a balloon to hold the extra blood; when blood volume decreases they contract somewhat. Pressure inside a compliant structure is determined by the volume and the compliance. From the central veins the blood re-enters the heart via the right atrium. It takes about a minute for the blood to circulate through the body.

After entering the heart via the right atrium, the blood goes to the right ventricle, then to the lungs, where it is re-oxygenated. After leaving the lungs, the blood goes to the left atrium, then to the left ventricle, from which it will exit the heart. As these interior divisions of the heart can be ignored for the purpose of studying the baroreceptor reflex, we can assume that the blood enters the heart via the right atrium and leaves immediately via the left ventricle.

The following definitions introduce the core parameters which CIRCSIM-Tutor is concerned with.

1) Heart rate (HR).
   Each heart beat allows some blood in and then forcefully ejects it. The number of beats per minute is the pulse or *heart rate*, abbreviated HR.

2) Inotropic state (IS).
   Inotropic state, or IS, is a measure of how much the heart contracts (or, similarly, how forcefully it contracts) with every beat. *Cardiac contractility*, abbreviated CC, is a synonym for IS which was used in version 2 of CIRCSIM-Tutor and in some of the human-to-human tutoring sessions.

3) Stroke volume (SV).
   The amount of blood pumped per beat is called the *stroke volume*, or SV. Stroke volume equals end-diastolic volume minus end-systolic volume. In other words, the difference between the volume of blood in the ventricle before contraction and the volume after contraction is the amount of blood ejected during the contraction.

4) Cardiac output (CO).
   Multiplying HR and SV gives the amount of blood pumped per minute, which is called *cardiac output* or CO.

5) Mean arterial pressure (MAP).
   Blood is pumped from the heart into the arteries, where the blood pressure is measured. The time average pressure of blood in the arteries is called the *mean arterial pressure* or MAP. The purpose of the baroreceptor reflex is to maintain a constant value for MAP.

6) Total peripheral resistance (TPR).
   From the arteries the blood flows through the arterioles. *Total peripheral resistance*, or TPR, is the resistance to blood flow through the circulatory system. Since arterioles are small in diameter and resist the flow of blood through them, arteriolar resistance is a major component of TPR.

7) Central venous pressure (CVP).
   From the arterioles, blood flows through a number of vessels until it

> reaches the central veins. Blood pressure in the venous system just outside the heart is called *central venous pressure*, or CVP. Central venous pressure is approximately equal to the pressure in the right atrium, known as *right atrial pressure* or RAP. Version 3 of CIRCSIM-Tutor uses CVP as one of the core variables in the model, but RAP was used in version 2 and in some of the human-to-human tutoring sessions.

These parameters are related as follows:

The equation CO = HR * SV states that cardiac output, which is the volume of blood exiting from the heart per minute, is a function of the rate at which the heart beats and the amount of blood ejected per beat. Thus we can say that an increase in HR or SV will cause an increase in CO.

When the heart pumps more blood per minute, more blood is forced into the arteries. Since the arterial system has resistance, pressure goes up. Thus increasing the blood flow per minute into the arteries causes an increase in arterial pressure. In other words, when CO rises, MAP rises.

TPR is another determinant of MAP. Another way to adjust blood pressure is to adjust the resistance of the arterioles. In contrast to the low resistance of arteries, the high degree of arteriolar resistance causes a marked drop in pressure as the blood flows through the arterioles. Arteriolar resistance is responsible for converting the pulsatile systolic-to-diastolic pressure swings in the arteries into non-fluctuating pressure in the capillaries. Instead of the elastic connective tissue in the arterial walls, arteriolar walls are made of smooth muscle richly innervated with sympathetic nerve fibers. Sympathetic nerve impulses cause the arterioles to contract, causing resistance to increase, which increases the pressure in the arteries which feed them. In other words, an increase in TPR causes an increase in MAP. In fact, there is an algebraic relation between MAP and its determinants: MAP = CO * TPR.

Note that the equations MAP = CO * TPR and CO = HR * SV are statements of causality as well as algebraic equalities. An increase in CO, for example, will cause an increase in MAP, but a change in MAP does not cause a direct change in the variables on the right-hand side. This is a frequent student misconception.

When cardiac output increases, increased quantities of blood are transferred from the venous system into the arterial system, decreasing the central blood volume and increasing the arterial blood volume (and pressure). The decrease in central blood volume causes central venous pressure (CVP) to go down. In other words, a increase in CO causes a decrease in CVP.

The central veins provide the input to the heart. When the central venous pressure increases, more blood flows into the heart on each beat. This quantity is known as *ventricular filling* or *preload*. An increase in ventricular filling causes the heart to beat more forcefully, a relationship known as the *Frank-Starling effect*. The cause of this phenomenon is the length-tension relationship, which states that the more muscle fibers are stretched, the more force they can develop. The more forceful beating of the heart causes more blood to be pushed out per beat, i.e. stroke volume is increased. In other words, an increase in CVP causes an increase in SV.

Inotropic state (IS) is another determinant of SV. The *ventricular function curve* shows the relation between ventricular filling and cardiac output, i.e. between the input and output of the heart. A change in IS causes the whole curve to shift up or down. In other words, at higher values of IS, the same degree of ventricular filling causes a higher stroke volume.

An increase in IS and an increase in ventricular filling both cause the heart to beat more forcefully, but through different mechanisms. Although an increase in ventricular

filling causes the heart to beat more forcefully, it does not cause IS to increase. This is another common student misconception.

Finally, an increase in mean arterial pressure causes SV to decrease. When mean arterial pressure increases, it becomes more difficult for blood to exit the heart. This causes stroke volume to decrease. The pressure that the ventricle has to pump against, i.e. the arterial pressure, is called *afterload*. This effect is less important than the other determinants of SV, and is in fact ignored by the older version of CIRCSIM-Tutor.

2.2.3 The role of the nervous system

The baroreceptor reflex attempts to control the value of MAP. When MAP rises above the level it should remain at, this rise triggers changes which force it down again. Similarly, when MAP falls too low, changes are triggered which force it back up.

The *baroreceptors* measure the blood pressure in the arteries, i.e. MAP. Although the baroreceptors are located in the neck, the arteries are large vessels which do not resist the flow of blood much, so the pressure measured by the baroreceptors is similar to the pressure at the point where the blood exits the heart.

The baroreceptors stimulate the *nervous system*, which then passes on messages about blood pressure to other organs of the body. The nervous system has two components, the *sympathetic* and *parasympathetic nervous systems*. The sympathetic nervous system acts as a kind of accelerator and the parasympathetic system as a brake. Usually they act in concert, so that these two aspects of the nervous system can be considered as one entity.

The variables directly controlled by the nervous system are called *neural variables*; the others are called *physical-chemical variables*. There are three neural variables:

HR: The nervous system controls HR by stimulating the sino-atrial node, which controls heart rate. When MAP goes up, HR goes down, so the heart beats fewer times per minute, reducing cardiac output, and thus reducing MAP.

TPR: The nervous system controls total peripheral resistance (TPR) by dilating or constricting the arterioles. The sympathetic tone supplied to the smooth musculature of the blood vessels by the nervous system determines their diameter. The diameter of a blood vessel most strongly determines its resistance to flow. When MAP goes up the blood vessels become wider, decreasing TPR, and thus reducing MAP.

IS: The nervous system controls IS by changing the intracellular concentration of calcium ions, which controls the force of contraction of the heart muscle. When MAP goes up, IS goes down, so less blood is pumped per beat (a decrease in stroke volume), reducing CO, and thus reducing MAP.

If the link from the baroreceptors to the nervous system is disabled, then the baroreceptors cannot affect the nervous system, so a change in MAP will not affect these variables. This operation is called *denervating* the baroreceptors.

2.2.4 Three stages: DR, RR, SS

In this section we introduce concepts necessary to understand the problems used by CIRCSIM-Tutor. Although knowledge about stages must be represented in any model, these particular definitions are specific to CIRCSIM-Tutor.

All negative feedback loops have a time component to their behavior; in the CIRCSIM-Tutor model, the body's response to a change, or *perturbation,* is divided into three stages:

- DR. The *direct response* or DR stage consists of those changes which happen immediately after the perturbation, before the baroreceptors are activated by the change in blood pressure.

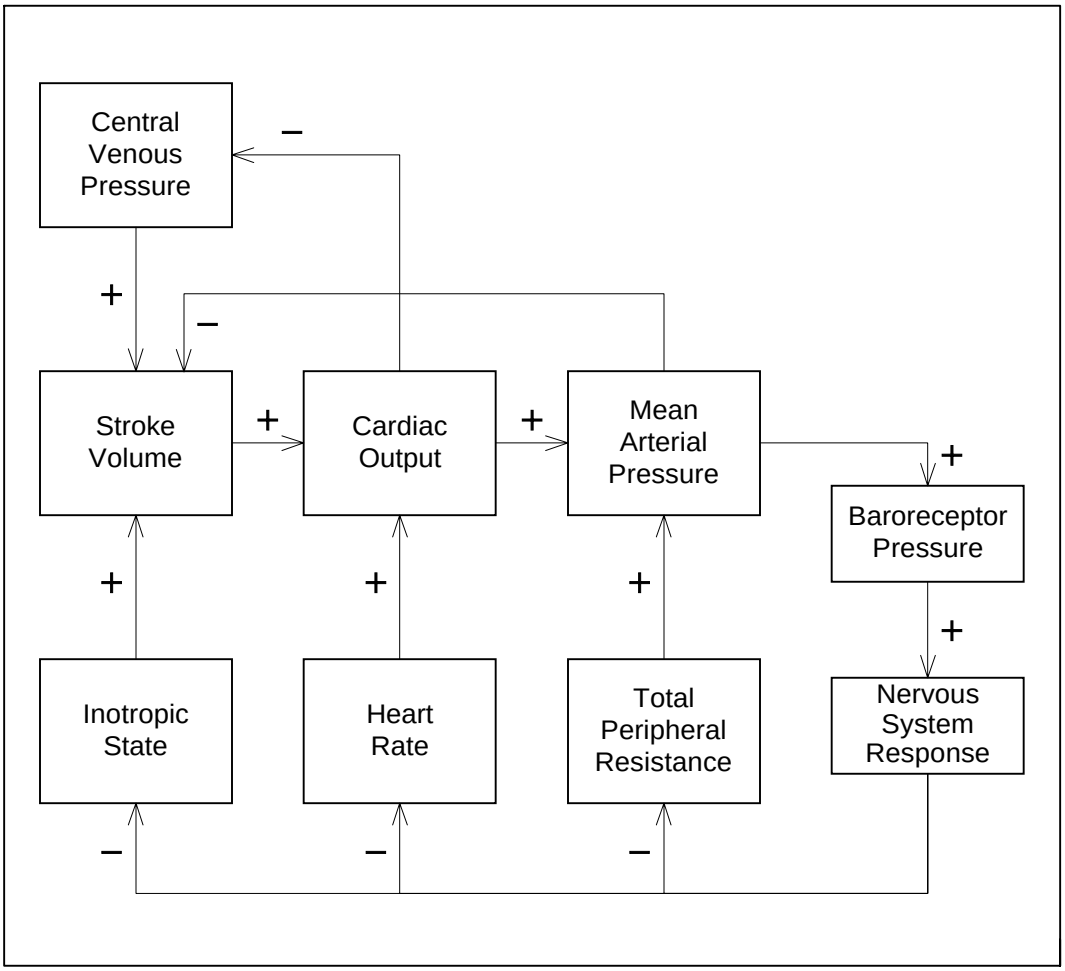

Figure 2.1: A version of the concept map

- RR. The *reflex response* or RR stage consists of changes that occur as a result of the activation of the baroreceptors, i.e. by the baroreceptor reflex. By convention, the values of parameters in RR are measured with respect to the value of the same parameters in DR, not with respect to the initial value before the perturbation.

- SS. The *steady state* or SS stage describes the state of the system after it has restabilized. Since, by convention, values in SS are measured relative to the initial value before the perturbation, the value of a parameter in SS will be the algebraic sum of its DR value and its RR value.

In the human body, a steady state is achieved within two to three minutes at most after the initial perturbation. The changes within a stage happen simultaneously but we model them for the student as a sequential process to help the student learn the causal reasoning involved.

2.2.5 Determinants and the concept map

The seven core variables and the relationships between them are summarized for the student on a *concept map*, a graphical memory aid shown in Figure 2.1. Students are given copies of the concept map and encouraged to use it to think about problems. The concept map does not uniquely define the causal model, as it does not indicate constraints on the relations, which relations take precedence, and whether changes happen simultaneously or sequentially. However, the concept map is a useful memory aid for students.

Each box in the concept map represents a parameter. An arrow with a plus sign indicates a direct relationship, i.e. when the first variable changes, the second variable changes in the same direction. An arrow with a minus sign indicates an inverse relationship, i.e. the second variable changes in the opposite direction. When two variables are related in this way, we say that A is a *determinant* of B. When a variable has multiple determinants, we need a way to decide which value should prevail, i.e. which variable is the *main determinant*. Since we are using a qualitative model, we cannot use look at the relative values in order to combine values. It is also convenient to define the concept of *minor determinant*, which is a variable whose value is only considered if none of the other variables has a value yet. The use of minor determinants does not add functionality to the model but it does simplify the representation.

The relationships described in Sections 2.2.2 and 2.2.3 can be summarized as follows:

Determinants:

HR, TPR and IS are neural.
The determinants of SV are CVP and IS, and MAP is a minor determinant (inverse direction).
The determinants of CO are SV and HR.
The determinants of MAP are CO and TPR.
CO is the sole determinant of CVP (inverse direction).

These relationships can be seen on the concept map or on Figure 2.3, which shows additional detail.

2.3 Defining the problem space

2.3.1 Defining the perturbations

After students have been introduced to the qualitative model of the heart, they are given problems to work. In each problem, a *perturbation* changes the processing of the heart. The student is then asked to predict the value (increase, decrease or no change) of the seven core variables in the DR, RR and SS stages, i.e. immediately after the perturbation, after the negative feedback loop has had time to operate, and after a new steady state has reasserted itself. Students store their responses on a tabular worksheet called a *prediction table* (Figure 2.2).

	DR	RR	SS
Central Venous Pressure			
Inotropic State			
Stroke Volume			
Heart Rate			
Cardiac Output			
Total Peripheral Resistance			
Mean Arterial Pressure			

Figure 2.2: Prediction table

There are two terms which belong to the tutor's model of the problem but are not used directly with the student. The *procedure variable* is defined as the first variable known to the system which is affected by the perturbation, and the *primary variable* is the first of the seven core variables to be affected. When talking to the student, our domain experts prefer to use paraphrases such as "first variable affected" and "first variable in the prediction table." If the procedure variable is one of the seven core variables, then the procedure variable and the primary variable are the same.

Perturbations can be caused by an event or by administering a drug to the patient. Sometimes we tell the student directly that a variable has changed without specifying the cause. The following list shows the principal events which CIRCSIM-Tutor can handle.

- Hemorrhage or transfusion
  A hemorrhage or transfusion causes a change in blood volume (BV). Because the central veins comprise the main blood reservoir of the body, any change in blood volume affects the central blood volume (CBV), i.e. the amount of blood in the central venous compartment. A hemorrhage causes the central blood volume to go down, causing a drop in central venous pressure (CVP). Similarly, a transfusion causes an increase in CVP.

- Centrifuge
  Putting a person in a centrifuge causes the central blood volume to drop by moving some blood out of the great veins toward the periphery of the body. Again, this causes central venous pressure (CVP) to drop.

- Broken pacemaker
  If a pacemaker which normally forces the heart to beat at 72 beats/min suddenly escalates to 120 beats/min, the heart rate has increased *ipso facto*.

Several kinds of drugs affect one or more of the neural variables. To understand the mechanism, one must look at the receptors of neural impulses. The signals sent out by the nervous system are received by three kinds of receptors: alpha receptors, beta receptors and cholinergic receptors.

- Alpha agonists and antagonists
  Alpha receptors are located on the walls of the arterioles. Thus drugs which affect alpha receptors, called *alpha agonists* or *alpha antagonists* based on whether they potentiate or inhibit the reception of signals by the alpha receptors, will affect the value of total peripheral resistance (TPR).

- Beta agonists and antagonists
  Beta receptors are located on the sino-atrial node, which controls the heart rate (HR), and on the heart muscle itself, which affects the value of inotropic state (IS). Thus drugs which influence the reception of signals by the beta receptors have two primary variables, HR and IS. For example, the commonly used beta-blockers are a type of beta antagonist. By blocking the reception of signals, they reduce the heart rate, and therefore cause blood pressure (i.e. MAP) to drop.

- Cholinergic agonists and antagonists
  Cholinergic receptors are also located on the sino-atrial node, so cholinergic agonists and antagonists cause an increase or decrease in HR respectively.

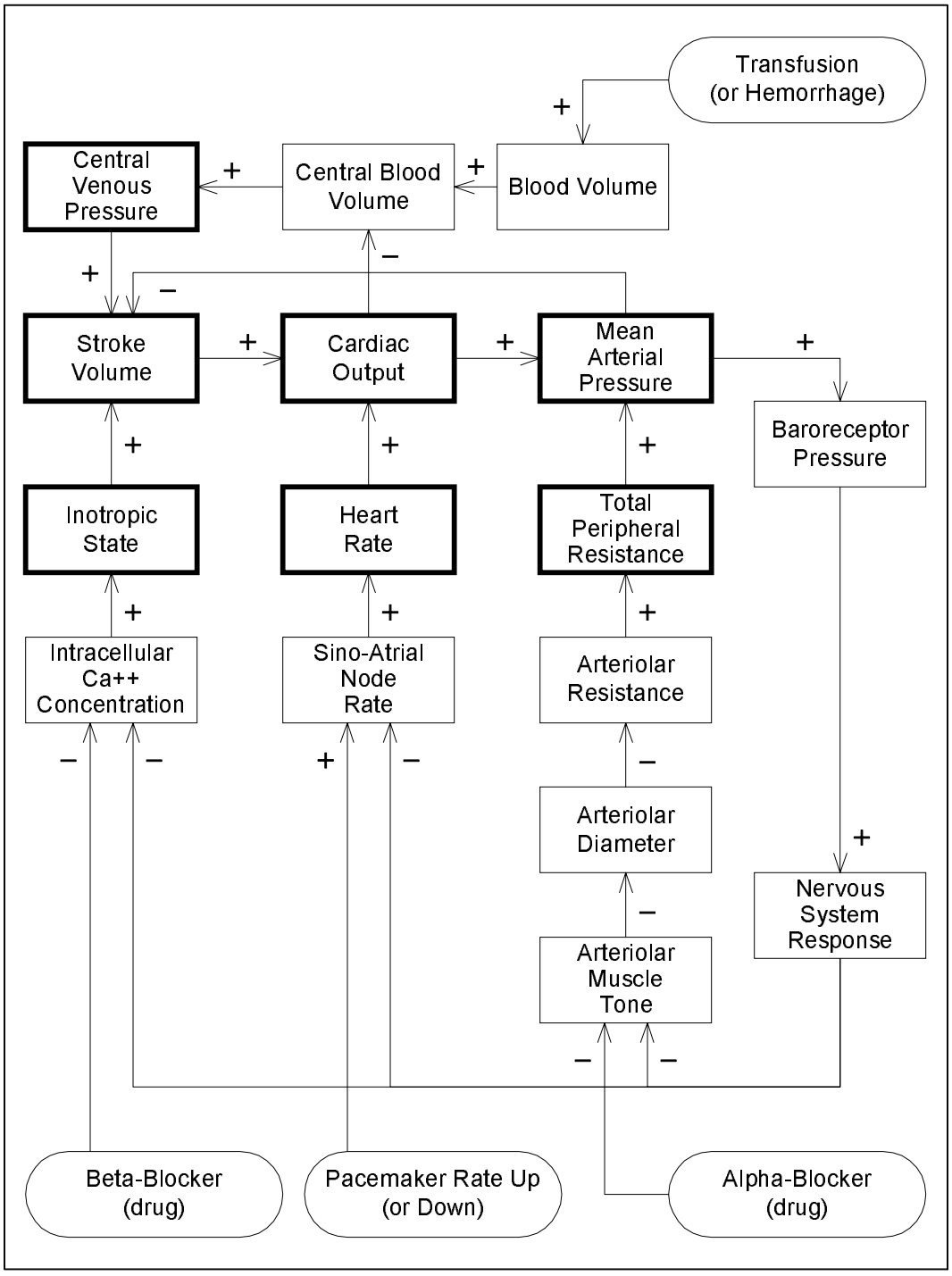

Figure 2.3: A more detailed concept map

In some cases, we tell the student directly that a variable has changed without specifying an external event which caused it.

- Change in arteriolar resistance (R_a)
  Since arteriolar resistance is the largest component of total peripheral resistance (TPR), an increase in arteriolar resistance will cause TPR to rise.

- Change in venous return (VR)
  Venous return is the rate at which blood returns to the central veins. An increase in venous return causes an increase in central blood volume, thus increasing central venous pressure (CVP).

- Change in intrathoracic pressure (P_{it})
  An increase in intrathoracic pressure reduces the space available for the central veins, thus causing an increase in central venous pressure (CVP). As we will see in Section 2.4.5, however, this case is different from other cases where CVP is the primary variable because an increase in P_{it} also causes SV to fall.

A chart summarizing a large number of perturbations is found in Appendix A.

2.3.2 Enumerating the simple problems

In this section we enumerate the possible problems containing one perturbation and one primary variable.

Studying the perturbations above, we see that only four of the seven core variables can be primary: central venous pressure (CVP), inotropic state (IS), heart rate (HR), and total peripheral resistance (TPR). The remaining three parameters can be affected only through their relationships to other variables. Once the primary variable and the direction in which it changes have been determined, the value of each variable in the DR stage is completely determined. Thus, from the point of view of DR, there are only four kinds of problems.

When the primary variable is neural, an additional fact is required to fully determine

the values for the RR stage. That factor is whether or not the primary variable continues to change in RR. For example, suppose a pacemaker malfunctions, causing the heart to beat at a constant but higher than normal pace. By definition, this means that heart rate has increased in the DR stage. Although the baroreceptors will send out signals to lower the heart rate in RR—that is, after all, the purpose of the system—the heart rate in such a patient is totally controlled by the artificial pacemaker and does not respond to the signals from the baroreceptors. We refer to such a variable as a *clamped variable*.

A different situation occurs when the change caused by the baroreceptor reflex can further update the change caused by the initial perturbation. For example, a hemorrhage will reduce the amount of blood in the central veins, causing central venous pressure (CVP) to decrease in DR. But the hemorrhage is now over: there is no reason why the baroreceptors cannot affect CVP in the RR stage. In fact, since the goal of the system is to maintain arterial blood pressure (MAP), it will move additional blood from the veins to the arteries, causing the counter-intuitive result that CVP will drop further in RR.

Sometimes the problem statement will tell the student whether a variable is clamped. For example, drugs can cause IS or TPR to be clamped if given in sufficient concentration. Heart rate (HR) is different because the sino-atrial node contains two kinds of receptors. Thus, even if one type is blocked, the other will not be.

Since each of the three neural primary variables can be clamped, the four simple cases in DR give rise to seven CIRCSIM-Tutor problems. Taking into account the fact that the initial stimulus in each problem can increase or decrease, there are 14 simple problems in total. Since the rules which determine the values of variables are symmetric with respect to “increase” and “decrease”, the solutions for each pair of related problems are inverses. The solutions to all of the simple problems are found in Appendix A.

2.3.3 Different ways of wording the problem for the student

Although there are only seven types of simple problems, each problem can be presented to the student in several ways. Khuwaja [1994] describes four levels of difficulty, based on the amount of inference needed to get from the immediate cause of the perturbation to the primary variable:

1) Identify the primary variable directly.
2) Identify the primary variable indirectly.
3) Directly identify a procedure variable other than the primary variable.
4) Indirectly identify a procedure variable other than the primary variable.

The increase in the difficulty level can be seen from the following examples:

1) Predict the effects of a drop in central venous pressure.
3) The patient lost a liter of blood due to hemorrhage.
4) The patient played several vigorous games of tennis without a break on a hot, humid day. Toward the end of the third game, the patient became dizzy and fainted.

2.4 Solving the problems

2.4.1 The language used in the rules

In this section we present rules for solving the problems. In later chapters we will discuss examples of dialogue which can be used to teach the rules. Although the text cannot be derived from the rules alone, different rules give rise to different textual formulations. The rules given in this section are largely mechanical, i.e. they deal with the propagation of values from one core variable to another based on rules which can be memorized or read from the concept map. Much of the dialogues conducted by expert tutors uses this type of language.

When talking about core variables is not sufficient to help the student, there are two ways to extend the language. Both of these methods use language similar to that in

Sections 2.2.2 and 2.2.3. In one case, we enrich the set of variables to include additional anatomical and physiological concepts. Within the CIRCSIM-Tutor project, this is known as using a deeper concept map. A further addition would be the addition of functional concepts. A functional concept is an abstract physical relationship which is not tied to a specific object, such as the relationship between volume and pressure. The current CIRCSIM-Tutor knowledge base does not currently contain knowledge about functional relationships; it can only store the relationship between a specific volume and its associated pressure.

2.4.2 Rules for the DR stage

Before giving the propagation rules for DR, we give the following rule, which applies in all stages:

Propagation-meta-rule:

Once a variable has been assigned a value in a stage, that value is never recomputed during that stage.

The algorithm for predicting the values for the DR stage is given below.

Determine-DR:

1. Determine the procedure variable and its direction from the perturbation.
2. Derive the primary variable and its direction from the procedure variable.
3. Determine the values of the neural variables.
4. Propagate values to variables on the shortest path to MAP.
5. Propagate values to the variables on the secondary path.

The first step in solving a problem is to determine the procedure variable. We also need to determine the *direction* in which the procedure variable changes, i.e. whether the perturbation causes it to increase or decrease. The next step is to determine the primary variable. For the cases we will be examining, both the procedure variable and the primary

variable can be determined from Appendix A or from Figure 2.1. By definition, the determination of the procedure variable is outside the range of the system, but the determination of the primary variable from the procedure variable can be done using the propagation rules we are about to introduce. We determine the procedure variable and the primary variable first because they can be determined without reference to other variables.

These rules for determining the procedure variable and the primary variable can be summarized as follows:

Det-procedure-variable:
When the value of the perturbation is propagated, the procedure variable is the first variable on the extended concept map to receive a value.

Det-primary-variable:
When the value of the procedure variable is propagated, the primary variable is the first core variable to receive a value.

The third step is to determine the values of the neural variables, i.e. heart rate (HR), total peripheral resistance (TPR), and inotropic state (IS). We determine the values of the neural variables next because they can be ascertained without reference to variables other than the primary variable. Since the neural variables are controlled by the nervous system and DR is the period before nervous system functioning is activated (i.e. before the changes being propagated through the system reach the baroreceptors), neural variables are not affected during the DR period. There is one exception to this rule. If the primary variable is neural, we already know from the previous step that it has changed. This is not a contradiction because it has not changed as a result of the baroreceptor reflex but because of a perturbation from outside the system.

So the following rule can be used to determine the value of the neural variables:

Neural-DR:

Neural variables which are not primary do not change in DR.

The need to teach this rule usually arises only in a situation where we already know that the variable is not primary. Thus we can teach it using the more concise formulation:

- Neural variables don't change in DR.

Although one rule is needed for deriving solutions and evaluating the correctness of student answers, the text we utter is somewhat different. This is one of the reasons that discourse schemata are necessary.

The fourth step is to determine the values of the variables along the shortest path to MAP. The *shortest path to MAP* is the shortest path from the primary variable to MAP. The shortest path to MAP must be stored as domain knowledge because there may be more than one path with the same length. It can be defined as follows:

Determine-shortest-path:

The shortest path is determined by the primary variable.

HR primary:	HR $\rightarrow$ CO $\rightarrow$ MAP
HR and IS primary:	HR $\rightarrow$ CO $\rightarrow$ MAP
No primary variable:	HR $\rightarrow$ CO $\rightarrow$ MAP
IS primary:	IS $\rightarrow$ SV $\rightarrow$ CO $\rightarrow$ MAP
TPR primary:	TPR $\rightarrow$ MAP
CVP primary:	CVP $\rightarrow$ SV $\rightarrow$ CO $\rightarrow$ MAP

Since the definition of the shortest path depends only on the primary variable, it is fully determined at this point. As we will see, the values of the variables on the primary path are now fully determined at this point also.

Several rules are needed to propagate values along a path in the concept map. For each variable on the shortest path, we determine its value by looking at the values of its determinants and whether each link inverts the direction of change. When there is only one

determinant, there is only one possible case:

Prop-1:

When a variable has one determinant, then the value of that determinant (increased, decreased or unchanged) determines the value of the variable.

These are the possible cases when there are two determinants:

Prop-2n:

(no value) If a variable has two determinants and one of them has a value and the other one does not have a value yet, then the value of the one with a value determines the value of the variable.

Prop-2u:

(unchanged value) If a variable has two determinants and one of them has increased or decreased and the other one is unchanged, then the value of the one which has changed determines the value of the variable.

Prop-2e:

(equal values) If a variable has two determinants and both of them have the same value, then that value becomes the value of the variable.

Prop-2sv:

(conflicting values for SV) In DR, if the two determinants of SV (i.e. IS and CVP) have conflicting values (one increases and the other decreases), the value of CVP determines the value of SV.

Rule *Prop-2sv* expresses the fact that CVP usually has a stronger effect on SV than IS. This results from the fact that HR, which determines CVP, is usually more powerful than IS.

When there are three determinants, i.e. two determinants plus a minor determinant, the following rule applies:

Prop-minor:

If a variable has two determinants and neither of them has a value other than "unchanged," but there is a minor determinant with a value, then the value of the minor determinant determines the value of the variable.

No other cases occur in DR, i.e. it never happens that there is one determinant but it doesn't have a value yet, or there are two determinants and both of them have values, or there are two determinants and neither of them has a value and either there is no minor determinant or it doesn't have a value either. Thus this set of rules uniquely determines the values on the shortest path to MAP in the DR stage.

The last step is to determine the values of the remaining variables. The *secondary path* contains all the variables that have not been assigned values after the shortest path to MAP has been traversed. Empirically these values always form a path. From a biological point of view, the secondary path contains the important link from CO to CVP unless CVP is the primary variable. The following rule can be used to determine the secondary path. No new rules are needed to determine the values along the secondary path.

Determine-secondary-path:

The secondary path is determined by the path which has already been traversed.

$HR \rightarrow CO \rightarrow MAP:$	CO → CVP → SV
$IS \rightarrow SV \rightarrow CO \rightarrow MAP:$	CO → CVP
$TPR \rightarrow MAP:$	MAP → SV → CO → CVP
$CVP \rightarrow SV \rightarrow CO \rightarrow MAP:$	(none)

We now know enough to complete the DR stage of an actual problem. The paragraph below contains the text of one of the problems which is used in the beginning physiology course.

> A patient with a non-functioning sino-atrial node has had an artificial pacemaker implanted. The pacemaker has been running at 72 beats/minute. Suddenly it malfunctions and the rate changes to 120 beats/minute.

We show below the result of applying these rules to the DR stage of this procedure. For each step, we give the rule which applies and one way to express it in English. The results

is an imaginary "think-aloud" session which represents the type of reasoning we want the student to internalize and the sequence we want the student to follow.

DR:

Procedure and primary variables:

HR: Since HR is the first variable known to CIRCSIM-Tutor to be affected by a faulty pacemaker, HR is the procedure variable. Since the pacemaker is beating too fast, HR will increase.
(Det-procedure-variable, perturbation = pacemaker)

Since HR is in the prediction table, it is the primary variable also.
(Det-primary-variable, procedure variable = HR)

Neural variables:

TPR: Since the DR stage is by definition the stage before the baroreceptor reflex is activated, a neural variable can't change in DR unless it's the primary variable. So TPR is unchanged.
(Neural-DR, V = TPR)

IS: By the same reasoning, IS is unchanged.
(Neural-DR, V = IS)

Shortest path to MAP:

CO: Since HR went up and SV has no value yet, CO (= HR * SV) must go up.
(Prop-2n, V = CO, $D_{changed}$ = HR, $D_{no\text{-}value}$ = SV)

MAP: CO going up will cause MAP to go up, since MAP = CO * TPR and TPR is unchanged.
(Prop-2u, V = MAP, $D_{changed}$ = CO, $D_{unchanged}$ = TPR)

Secondary path:

CVP: CO going up will cause CVP to go down.
(Prop-1, V = CVP, D = CO)

SV: Since CVP went down and IS is unchanged, SV will decrease.
(Prop-2u, V = SV, $D_{changed}$ = CVP, $D_{unchanged}$ = IS)

2.4.3 Rules for the RR stage

In the RR stage, the baroreceptors have responded to the change in MAP in the DR stage. They activate the nervous system, which sends signals to various parts of the body. These signals cause a change in any neural variable whose value is not clamped. Referring to Section 2.2.3, we see that a change in MAP in DR causes neural variables to move in the opposite direction in RR.

The following propagation rules can be used to determine the values of the variables in RR.

Determine-RR:

1. Determine values for the neural variables.
2. Propagate values to variables on the most important path to MAP.
3. Propagate values to variables on the secondary path.

The following two rules can be used to determine values for the neural variables.

Neural-RR:

If MAP went up in DR, then any non-clamped neural variable will go down in RR, and vice versa.

Clamped-RR:

Clamped neural variables do not change in RR.

The *most important path to MAP* is the path where the most significant changes take place. This path starts with HR unless HR is clamped. It can be determined as follows:

Determine-most-important-path:

No clamped variable:	HR $\rightarrow$ CO $\rightarrow$ MAP
IS clamped:	HR $\rightarrow$ CO $\rightarrow$ MAP
TPR clamped:	HR $\rightarrow$ CO $\rightarrow$ MAP
HR clamped:	IS $\rightarrow$ SV $\rightarrow$ CO $\rightarrow$ MAP
HR and IS clamped:	TPR $\rightarrow$ MAP

The secondary path in RR is determined using the same rule as in DR.

The following solution trace continues the previous example.

RR:

Neural variables:

HR: The pacemaker prevents the nervous system from having any effect on HR, so HR is unchanged from its DR value.
(Clamped-RR, V = HR)

IS: Since MAP went up in DR, IS will decrease.
(Neural-RR, V = IS)

TPR: By the same reasoning, TPR will decrease.
(Neural-RR, V = TPR)

Most important path to MAP:

SV: Since IS decreased and CVP has no value yet, SV will go down.
(Prop-2n, V = SV, $D_{changed}$ = IS, $D_{no\text{-}value}$ = CVP)

CO: Since SV went down and HR is unchanged, CO will go down.
(Prop-2u, V = CO, $D_{changed}$ = SV, $D_{unchanged}$ = HR)

MAP: Since CO and TPR both decreased, MAP will go down.
(Prop-2e, V = MAP, D_1 = CO, D_2 = TPR)

Secondary path:

CVP: CO going down will cause CVP to go up because CO inversely affects CVP.
(Prop-1, V = CVP, D = CO)

2.4.4 Rules for the SS stage

In SS, the system returns to a steady state. The following propagation rules can be used to determine the values of the variables in SS.

Determine-SS:

1. Determine the value of the primary variable.
2. Determine values for the remaining neural variables.
3. Propagate values to variables on the shortest path to MAP.
4. Propagate values to variables on the secondary path.

Neural variables have done all the changing they are going to do in RR, so they have no further changes to make to come to a steady state in SS. But for primary variables, the majority of the change took place in DR. Thus we have the following rules for primary variables, whether neural or not, and for other neural variables in SS:

Primary-SS:
If a variable is primary, it has the same value in SS as it did in DR.

Neural-SS:
If a neural variable is not primary, it has the same value in SS as it did in RR.

If MAP decreases in DR and thus increases in RR, it will usually still be decreased in SS relative to its initial value. This is an important principle which is usually expressed to the student as follows:

- The reflex never fully compensates [for the perturbation].

This is a basic principle of negative feedback systems. For example, suppose we give a patient a transfusion. This will cause MAP to increase and thus cause the values of the neural variables to decrease. In the new steady state, MAP will remain elevated and the neural variables decreased until the extra fluid is excreted.

This gives us the following rule:

Compensate-MAP:
The value of MAP in SS is the same as its value in DR.

We could work backward from MAP to obtain values for the other variables. However, we prefer to work forward to be consistent with the previous stages. Thus we do not use *Compensate-MAP* in our solutions.

The following rule can be used along with the rules for primary and neural variables given above to solve SS without using any propagation rules.

Algebraic-SS:

The value of a variable in SS is the "algebraic sum" of its qualitative predictions in DR and RR. If the DR and RR values have opposite signs, the DR value prevails.

This rule is simple and easy to explain. However, because the propagation rules may give a deeper understanding of the mechanism, we complete our example by giving the solution trace for SS using the propagation rules.

One additional rule is needed to complete propagation in SS. The intent of the following rule is to permit propagation along the path SV $\rightarrow$ CO $\rightarrow$ MAP even when there is conflicting data from HR and TPR.

Prop-2c:

(conflicting determinants for CO and MAP) If a variable other than SV has two determinants with conflicting values, and one determinant is neural and the other is not, then the non-neural determinant takes precedence unless the neural determinant is primary.

In fact, rule *Prop-2c* can be used to replace all of the forms of *Prop-2* given above except for *Prop-2sv*. The clause "unless the neural determinant is primary" does not apply in any of the current cases but is necessary if *Prop-2c* is used to replace other forms of *Prop-2*. An advantage of rule *Prop-2c* is that it makes the derivations independent of the sequence in which they are done.

SS:

Neural variables:

HR: Since HR is primary, it has the same value in SS as in DR, so it rises.
(Primary-SS, V = HR)

TPR: Since TPR is not primary, it has the same value in SS as in RR, so it decreases.
(Neural-SS, V = TPR)

IS: Since IS is not primary, it has the same value in SS as in RR, so it decreases.
(Neural-SS, V = IS)

Shortest path to MAP:

CO: Since HR went up and SV has no value yet, CO (= HR * SV) must go up.
(Prop-2n, V = CO, $D_{changed}$ = HR, $D_{no\text{-}value}$ = SV)

MAP: CO went up and TPR went down. CO overrides TPR because TPR isn't primary, so MAP goes up.
(Prop-2c, V = MAP, $D_{non\text{-}neural}$ = CO, D_{neural} = TPR)

Secondary path:

CVP: CO going up will cause CVP to go down.
(Prop-1, V = CVP, D = CO)

SV: Since both CVP and IS went down, SV will decrease.
(Prop-2e, V = SV, D_1 = CVP, D_2 = IS)

2.4.5 Multiple primary variables, multiple perturbations, and other special cases

There are several ways to create more complex problems, and additional rules are needed to solve them.

- *Multiple primary variables*. For example, beta-blockers, a class of drugs used to reduce blood pressure, affect both heart rate and inotropic state.
- *Exceptions*. Although the rules given above are a useful model, sometimes instructors would like to use a problem which requires a more complex model.
- *Denervating the baroreceptors*. If the connection between the baroreceptors and the nervous system is severed, additional rules are required to describe the resulting behavior of the system.
- *Multiple perturbations*. The student can be asked to predict the effect of sequential perturbations. For example, a pacemaker can break after a drug has been administered to the patient.

Beta-blockers have two primary variables, HR and IS. The sino-atrial node, which controls heart rate, contains both beta receptors and cholinergic receptors. Since beta-blockers block only the effect of the nervous system on beta receptors (i.e. through the sympathetic nervous system) and not on cholinergic receptors (i.e. through the parasympathetic nervous system), HR will not be clamped. IS will be clamped if a sufficient quantity of the drug is given. No additional rules are needed to solve problems involving beta-blockers. A complete solution is shown in Section A.8.1.

A change in intrathoracic pressure (P_{it}) is an example of a perturbation which requires rules more complex than those given above. When intrathoracic pressure (P_{it}) increases, all of the structures inside the chest are compressed, including the central veins and the chambers of the heart. Since the central veins are compressed, pressure inside them rises, so the primary variable is central venous pressure (CVP), which increases. Normally an increase in CVP would cause more blood to flow into the heart on each beat, i.e. an increase in ventricular filling. This would cause more blood to flow out per beat also, i.e. stroke volume (SV) would increase. But when P_{it} increases, the arteries where the blood will go as it leaves the heart are compressed also. As a result, ventricular filling decreases, causing a decrease in SV. The following enhancement to rule *Prop-1* is needed to solve problems involving P_{it}. Section A.8.2 contains the details of the solution.

Prop-1-Pit:

> The following case is an exception to rule *Prop-1*. When intrathoracic pressure (P_{it}) is the primary variable, SV is inversely determined by CVP.

Denervating the baroreceptors means to cut the link from the baroreceptors to the nervous system. Since the nervous system is not active in DR, this means that the effect of the perturbation is not noticed in DR, i.e. no variables change. In RR, the complete

cessation of signals from the nervous system will appear to the receptors on the effector organs as a very steep fall in MAP. As a result all of the neural variables will rise in RR, and none is clamped. The following two rules describe this situation. A complete solution is shown in Section A.8.3.

Denervate-DR:

If the baroreceptors have been denervated, then no variables change their values in DR.

Denervate-RR:

If the baroreceptors have been denervated, then all the neural variables rise in RR.

When two perturbations occur in sequence, we process them sequentially and illustrate the result by showing the prediction table after each perturbation. The DR for the second procedure is based on the result of SS in the first procedure. In other words, DR for the second procedure measures the change from the previous steady state. This result differs from the regular DR in only one case. When the first perturbation causes a variable to be clamped, it stays clamped in RR during the second procedure, with the exception that a broken pacemaker in the second procedure can override the result of clamping by a drug. The following rules are needed to solve the multiple-perturbation problems given in Sections A.8.4 and A.8.5. It is possible that other problems involving multiple perturbations would require additional rules.

Clamped-RR-previous:

If a variable is clamped during the first perturbation of a multiple-perturbation problem, it does not change in RR of the second step except in the case where the variable in question is HR and the second procedure involves a broken pacemaker.

Denervate-RR-longterm:

If the first procedure involves denervating the baroreceptors, then no variables change in RR in the second procedure.

The intent of *Clamped-RR-previous* is to express the idea that a clamped variable stays clamped. Since an artificial pacemaker acts directly on the sino-atrial node, it takes precedence over any other influence.

Denervate-RR-longterm expresses the idea that although cutting the link from the baroreceptors causes the reaction described by *Denervate-RR*, once that link is cut, there is no way to trigger a further reaction.

As the tutoring sessions from which we abstracted our pedagogical and linguistic knowledge did not contain any examples of these complex cases, some additional schemata might be required to handle them correctly. For example, to handle multiple perturbations one would need a schema which went through the stages twice. However, it is unlikely that enhancements to the lower-level aspects of text generation would be required. The ability to add these additional cases quickly and easily is a consequence not only of the use of artificial intelligence techniques but of the modular design of the system.

Chapter 3
Introduction to the CIRCSIM-Tutor Project

Increasingly, people seem to misinterpret complexity as sophistication, which is baffling—the incomprehensible should cause suspicion rather than admiration. Possibly this trend results from a mistaken belief that using a somewhat mysterious device confers an aura of power on the user. —*Niklaus Wirth*

The previous chapter introduced the physiological concepts necessary to understand a CIRCSIM-Tutor problem and its solution. In this chapter we examine how an automated system can be used to tutor students on the solution to these problems. We start by examining earlier CAI programs for tutoring cardiovascular physiology. Then we review the history of CIRCSIM-Tutor in order to motivate the research goals listed in the Introduction. Finally, we describe two aspects of the student's interaction with CIRCSIM-Tutor: the user interface, which determines how the student interacts with the program, and the experimental protocol, which determines the overall flow of the student's session by controlling when the tutor can intervene in the prediction process.

3.1 Computer-assisted instruction (CAI) systems for the baroreceptor reflex

3.1.1 MACMAN: A quantitative simulation

Michael and Rovick were first introduced to the possibilities of using the computer as a teaching aid through MACMAN [Dickinson, Goldsmith & Sackett 1973], a mathematical model of the baroreceptor reflex developed at McMaster University. MACMAN was a FORTRAN program which ran on a mainframe and printed its output on a

line printer. It had a simple interface that allowed the user to specify the parameters for running the model and to designate which parameters would be printed, either graphically or in tabular form.

Rovick and Michael translated the program into TUTOR, the language for the PLATO system. They used it with a lab manual that described experiments to be carried out, allowed students to design and carry out their own experiments, and included questions to direct the students' attention to relevant phenomena.

After a year of experimentation, they concluded that MACMAN was not particularly useful to their students. The students tended to do the experiments in a rote fashion without thinking about the implications of the results. Additionally, they did not know enough about physiology or experimental design to generate their own experiments. Thus the success of MACMAN was totally dependent on interaction with an instructor.

3.1.2 HEARTSIM: Adding a didactic component

Rovick and Michael believed that the reason for the failure of MACMAN was the fact that it was purely a simulation with no pedagogical component. Their response was to develop HEARTSIM [Rovick & Brenner 1983], a PLATO program which included a didactic component in addition to MACMAN.

The didactic component introduced three new features which proved to be successful in helping students learn and which they have therefore carried through to their more recent systems. The three features are:

- Reliance on qualitative instead of quantitative predictions
- Use of the prediction table as an interface device
- Predefined set of problems for the students to work

Rovick and Michael switched to the use of qualitative predictions after realizing that they

would be satisfied if the student could make a qualitative prediction, i.e. state whether the value of each variable would increase, decrease, or remain unchanged. They introduced the prediction table [Rovick & Michael 1992], a chart with spaces for the student to enter a qualitative prediction for each variable at each physiological stage, which was illustrated in Figure 2.2. (The prediction table is also shown in Figure 3.1 as part of the graphical user interface for CIRCSIM-Tutor.)

The prediction table served three purposes:

- Give the students a way to organize their thinking
- Provide an easy way for students to enter data
- Provide a simple way for the system to give feedback (by highlighting errors)

In addition to switching from quantitative to qualitative predictions, Rovick and Michael switched from having the students specify the parameters they wanted to modify to providing a menu of predefined problems. Although HEARTSIM continued to give students the opportunity to design and run their own experiments, relatively few students used this feature of the program.

Students used HEARTSIM with the protocol given below. While the quantitative and graphical results were provided by MACMAN, the qualitative results used for the comparison were stored rather than calculated.

1. Select a problem.
2. Make predictions for all three stages.
3. Correct errors.
4. See the results of the procedure plotted and displayed as a table.
5. Compare predictions with the actual results and study canned text selected by the system.

3.1.3 CIRCSIM: CAI program using qualitative reasoning

CIRCSIM is a BASIC program for DOS machines whose development was prompted by the limited availability of PLATO. In developing CIRCSIM, Michael and Rovick realized that the mathematical model in HEARTSIM was used only to display graphs showing the behavior of the variables over time. Since in their opinion most of the learning was being generated from the qualitative stored predictions rather than the quantitative information, they did not give CIRCSIM a mathematical model. Instead CIRCSIM stores the correct predictions and the limited data needed to display the desired graphs.

To provide more guidance to the student, CIRCSIM contains a procedure which can lead a student through the solution to a problem. This procedure also helps the student develop a general model for negative feedback systems and a method for solving the problems. CIRCSIM currently has over 250 canned text paragraphs which it can use for tutoring.

3.2 Comparison of v. 3 of CIRCSIM-Tutor to v. 2

3.2.1 Motivation for the development of CIRCSIM-Tutor

Parallel to the development of CIRCSIM, Michael and Rovick also realized that while CIRCSIM was an effective learning tool, there were many kinds of errors and misconceptions that it was unable to remedy. This conviction arose from the observation that students continued to have significant problems with their understanding of the baroreceptor reflex even after using CIRCSIM. They believed that the solution to this problem would be the availability of a program that could conduct a dialogue with the student. Several versions of CIRCSIM-Tutor have been developed under the supervision of Professor Martha W. Evens of the Illinois Institute of Technology

The current version of CIRCSIM-Tutor is v. 2, developed by Woo [1991] and others using Procyon Lisp for the Macintosh. A prototype had earlier been implemented in Prolog by Kim [1989]. Many of the high-level aspects of v. 2 fit the problem well and have been carried over into v. 3. On the other hand, v. 3 is more sophisticated than v. 2 in many respects, including planning, dialogue handling, and text generation. In some situations, v. 3 generalizes a feature available in v. 2 only in specific situations which are hard-wired into the program. The crucial difference between v. 2 and v. 3 is that the primary goal of v. 3 is to generate a dialogue. With text generation as a primary goal, it is easier to generate the variety of tutoring phenomena used by our expert tutors while maintaining a coherent conversation. Pedagogical planning rules are still used to decide which tutoring method to use at any point in time, but the output of these rules is a plan for a piece of text which will be integrated into the developing conversation. As a result discourse goals generated by disparate pedagogical goals can be amalgamated into one turn or even into a single sentence. Textual mechanisms needed for ensuring coherence (*so*, *but*, etc.), for switching topics or for handling a student initiative can be generated without disturbing the pedagogical plan.

Although Zhang's [1991] work precedes v. 2, it contains the crucial idea that a text model separate from the pedagogical model is needed for text generation. However, Zhang's text planner uses descriptive terms such as *hint* as goals. In our view, as demonstrated in Sections 4.6 and 4.11, hints and other descriptive phenomena are created as the result of one or more subgoals of the text planner.

3.2.2 Features and shortcomings of CIRCSIM-Tutor v. 2

The following list shows some of the main features of v. 2.

- *User interface.* The handling of the user interface and experimental protocol in v. 2 fit the problem well and will be carried over into v. 3, although the default protocol is different. Students use a prediction table to make their predictions, then engage in a tutorial dialogue led by the tutor.

- *Top-level dialogue organization.* Version 2 tutors the student on each incorrect prediction. Within each stage, the variables are tutored in causal sequence, i.e. in the sequence in which they are encountered while solving the problem.

- *Tutoring methods.* Version 2 has two principal methods for tutoring variables, one for neural variables and one for non-neural variables. Each of these methods uses a multi-turn plan. When the student gets multiple neural variables wrong, a shorter correction method is used for the second and subsequent variables.

- *Turn structure and responding to the student.* Version 2 always responds to the student before continuing with the tutor's plan. When the student makes a mistake, v. 2 chooses whether to give the correct answer or to give the student a hint first. When two determinants are requested and only one is given, v. 2 can prompt for the second determinant.

- *Text generation.* Each turn consists of one or more syntactically correct sentences. Many are generated from semantic forms; some of the more complex ones are hard-coded.

Viewed from today's perspective, however, v. 2 has many shortcomings:

Planning:

- *Inability to generate nested text structures.* Since v. 2 generates text by means of an augmented finite-state machine, it cannot generate nested structures unless they are hard-wired in the finite-state machine. One hierarchical text structure which is hard-coded in the finite-state machine is the correction of variables within stages. Version 3 can handle arbitrary nested structures.

- *Lack of retry capability.* Version 2 does not have the general capability to retry a goal which has failed. It can only retry a goal when an alternative response has been pre-programmed. Otherwise the only option is to give the student the correct answer. Two kinds of retry, the ability to generate a hint ("Consider ...") in certain contexts and the ability to ask for a second determinant when only one is given, are hard-coded into the finite state machine. Version 3 can retry any goal as long as an alternative is available.

- *Inability to drop a plan.* Version 2 cannot drop a multi-turn method if the student's response is disappointing. Version 3 can change plans whenever an alternative is available.

  Chapter 5 contains a detailed account of the planning process for v. 3.

Turn structure:

- *Restricted turn structure.* Version 2 always gives the student an explicit acknowledgment, either interpersonal (e.g. *correct*), content-based (*SV is the correct answer*), or both. But human tutors often leave the acknowledgment implicit. Additionally, in v. 2 the response to the student is always a declarative statement, never a question or a multi-turn plan. As shown in Section 4.5, version 3 can use any type of tutorial method to reply to the student. It can also leave the acknowledgment explicit when desired.

Coherence:

- *Insufficient attention to intra-turn coherence.* Version 2 generates each sentence of a turn individually, causing turns to be more or less coherent depending on which path through the finite-state machine was chosen. As a result text generated by v. 2 has the stylized, stilted quality we associate with compiler error messages rather than with language used in natural situations. Since each discourse goal generates a separate sentence, text generated by v. 2 does not contain subordinate clauses. Since it does not attempt to connect each sentence with earlier sentences in the turn, methods of cohesion such as pronouns are never generated. In order to avoid these problems, v. 3 generates text for each turn as a whole.

- *Insufficient attention to inter-turn coherence.* Coherence of the conversation as a whole, i.e. inter-turn coherence, is less of a problem but is still an issue. Text for tutoring each variable is generated individually. Since there is only one schema for tutoring each type of variable and each of them has content-based coherence, large-scale coherence largely takes care of itself. However, in the next section we will see several examples where inter-turn coherence can be improved through the use of discourse markers. Creating inter-turn and intra-turn coherence in v. 3 is discussed further in Section 5.4.5.
- *Unnecessary repetition in generated text.* The use of the *correct-non-neural* schema illustrated in the next section causes significant repetition. Each non-neural variable is treated as a separate unit. The system asks for the determinants of each variable even though that information may have been given by the student earlier in the conversation. This behavior is significantly different from that employed by human tutors, who invite the student to build on the results for each variable in order to solve the next. In contrast, v. 3 contains a *move-forward* schema, described in Section 4.7.2, which enables the tutor to base the value of a variable on a previous one.

Domain model:

- *Inability to go beyond the core variables.* In v. 2, the domain model, the pedagogical model and the text schemata contain information solely about the seven core variables. The knowledge base in v. 3 contains sufficient information to generate text about causal relationships on a deeper concept map, although it does not contain functional information.

Variety (pedagogical, linguistic and lexical):

- *Pedagogical variety.* The fact that there is only one schema for each type of variable makes it too easy for the student to respond automatically without thinking about what the tutor is saying. Furthermore, work done by the CIRCSIM-Tutor project on creating a more sophisticated student model is less useful without a selection of alternatives for the planner to choose from.

- *Syntactic variety.* Most discourse goals in v. 2 have only one potential realization. Repetition at the pedagogical and syntactic

levels is the primary reason why CIRCSIM-Tutor v. 2 does not always give the feel of communicating with an intelligent agent. But adding alternative realizations without a mechanism to ensure coherence will not lead to better text.

- *Lexical variety.* Although v. 2 can choose multiple lexical items for some concepts, it does so randomly. Since people make their lexical choices based on pragmatic considerations, in some cases random lexical choice sounds less natural than simple repetition would. The choice of syntax and lexical items in v. 3 is discussed further in Section 5.4.3.

3.2.3 Sample texts from CIRCSIM-Tutor v. 2

This section illustrates the features and weaknesses of v. 2 described in the previous section. Version 2 has two major teaching plans, one for neural variables and one for non-neural variables. Using the notation to be introduced in Section 4.6.1, the plan for correcting neural variables can be described with the following schema.

Correct-neural (?v):
    Teach-info (has-definition(DR, definition-of(DR)))
    Elicit-info (has-mechanism(?v, neural))
    Elicit-info (has-value(?v, DR, no-change))

Shorter versions are provided for the second and third incorrect neural variables:

Correct-second-neural (?v):
    Elicit-info (has-mechanism(?v, neural))
    Elicit-info (has-value(?v, DR, no-change))

Correct-third-neural (?v):
    Elicit-info (has-value(?v, DR, no-change))

As these schemata are closely aligned with the methods used by expert human tutors in our transcripts, they will be reused in v. 3, albeit with more varied realizations (see Section 4.6). The following three-part example gives the flavor of v. 2 while illustrating the three versions of the neural paradigm.

(1) T: Remember that the Direct Response occurs immediately and produces physical changes in the system before any reflex is activated. What controls HR *(heart rate)*?
S: Nervous system.
T: Correct. Heart rate is neurally controlled. What is the correct value of HR?
S: Unchanged.
T: Correct, the value of HR is unchanged. Please redo the variable HR and click on the "done" button.
S: *(changes prediction table)*

(2) T: What controls CC *(cardiac contractility, i.e. inotropic state)*?
S: Nervous system.
T: Great, cardiac contractility is controlled by the autonomic nervous system. What is the correct value of CC?
S: Unchanged.
T: Please redo the variable CC and click on the "done" button.
S: *(changes prediction table)*

(3) T: What is the correct value of TPR *(total peripheral response)*?
S: Unchanged.
T: Please redo the variable TPR and click on the "done" button.
S: *(changes prediction table)*

This text is accurate but repetitive. It would read more smoothly if some of the explicit acknowledgments were omitted.

The following schema illustrates the pattern which v. 2 uses to correct any non-neural variable. (This pattern exists in v. 3 but is relatively uncommon; see Section 4.7.1).

Correct-non-neural (?v):
Get-determinants (?v)
Get-main-determinant (?v, ?determinant)
Get-relationship (?v, ?determinant)
Get-value (?v)

The following excerpt shows a typical instantiation of this schema. It also shows an example of one of the typical error correction paths in v. 2. In turn 2, the student gives an incorrect answer. The tutor's reaction in turn 3 contains three parts. First the tutor gives a

content-based reply ("MAP is the wrong answer"). Then the tutor satisfies the *get-determinants* subgoal by giving the answer ("SV is determined by …"). Finally, the tutor moves on to *get-main-determinant*, which is the next subgoal.

(4)
1 T: What are the determinants of SV?
2 S: MAP.
3 T: MAP is the wrong answer. SV is determined by RAP and CC. What is the important determinant in this case?
4 S: RAP.
5 T: Correct, the important determinant of SV in this case is RAP. What is the relation between RAP and SV?
6 S: Direct.
7 T: Good, you got the correct answer. What is the correct value of SV?
8 S: Decrease.
9 T: Correct, the value of SV is decreased.

This text is grammatically acceptable but stilted. Again, it would read more smoothly if some of the explicit acknowledgments were omitted. Because v. 2 plans text independently for each turn without taking previous turns into account, the generated text occasionally lacks *inter-turn coherence*. For example, turn 7 could be expressed more smoothly as follows:

(5) T: Correct. So what is the value of SV?

The discourse marker "so" makes this text sound more natural because it reminds the reader that a subtopic has been completed and that the tutor is returning to the previous topic [Schiffrin 1987]. Because v. 2 generates text for each turn independently of the evolving conversation, it never generates such cues for the reader.

From a discourse functional point of view, the problem is that each sentence uttered by the tutor expresses a proposition with all arguments fully spelled out. In normal human discourse, people leave some parts of most propositions implicit. The fully spelled-out propositions accentuate the related problem of unexpected topic shifts, such as from MAP

to SV in turn 3. These issues could be summarized as a lack of *intra-turn coherence*. The examples below show some more natural replies which the tutor could give for turn 3. Each of them expresses the same ideas as the original, but in shorter and more natural-sounding language. In the suggested replacements, several linguistic mechanisms are used to increase coherence: sentences have been combined, pronouns have been inserted where appropriate, and some ideas have been left implicit.

(6) T: No, SV is determined by RAP and CC.
Now which one of those is the important one in this case?

(7) T: No, the determinants of SV are RAP and CC.
Which one is relevant here?

(8) T: No, the right answer is RAP and CC.
Which one is most important in this case?

The issues of intra-turn coherence and syntactic variety are closely associated. Although it would be easy to add more lexical items and sentence structures to v. 2, that would just create more cases which need to be tweaked in order to produce natural-sounding text. The underlying issue is that v. 2 generates text for each pedagogical goal independently. We need a way to take into account that several goals could be uttered together in one turn without having to provide a template for every combination of goals.

The following two excerpts show hints generated by v. 2. Although the hints are structurally identical, the first example is more coherent than the second. In the first example, all of the sentences in the tutor's response deal with the values of variables, and RAP is a reasonable variable to consider in order to determine the value of SV. In the second example, the topic changes from determinants to variables and back again. Although the two examples are different, both would read more smoothly if a discourse marker were added to the last sentence, e.g. *Now what is the value of SV?*

(9) T: ... What is the correct value of SV?
S: Up.
T: No, SV does not increase. Consider the value of RAP. What is the correct value of SV?

(10) T: What are the determinants of RAP?
S: CC.
T: Sorry. RAP is not determined by CC. Consider the value of CO. What are the determinants of RAP?

The following example shows an example of lexical choice where v. 2 does not take pragmatic considerations into account. The tutor must believe that the student knows the meaning of the abbreviation CC in order to use it in the first turn. The student's response does nothing to contradict this belief. Thus it sounds odd when the tutor uses the term *cardiac contractility* in the following turn. A similar case occurs in the third turn of example (2).

(11) T: The value of a neural variable in SS is the same as its value in RR because neural variables don't change in DR except when one is a primary variable. What is the value of CC in DR?
S: Decrease.
T: Correct, the value of cardiac contractility is decreased in DR.

Although both CIRCSIM and CIRCSIM-Tutor v. 2 are useful learning aids for students, each of them is limited in various ways. In fact, students who have used both sometimes feel that CIRCSIM is more intelligent because the canned text it uses is both linguistically and physiologically more sophisticated. There is no contradiction here; it is similar to saying that in a specific range an algorithm with an $O(n^2)$ running time can be faster than one with an $O(n)$ running time. The goal of v. 3 is to generate more coherent text while at the same time expanding the range of situations which the system can discuss and the ways it has to discuss them. The goals stated for v. 3 in the Introduction were derived from this analysis of the shortcomings of v. 2 as well as from the analysis of the productions of human tutors described in Chapter 4.

3.3 User view of CIRCSIM-Tutor v. 3

3.3.1 User interface for CIRCSIM-Tutor v. 3

CIRCSIM-Tutor v. 3 is written in a Lisp compatible with both Allegro Common Lisp for Windows and Macintosh Common Lisp. Only the screen handling code is different for the two systems. The basic user interface is the screen shown in Figure 3.1. The left-hand side contains a prediction table where the student can fill in predictions for the DR, RR and SS stages. The right-hand side of the screen contains a window where dialogue with CIRCSIM-Tutor will unfold. A small window at the top of the screen contains a summary of the current problem for the student's reference. The student can enlarge this window to see the complete problem description if desired. At the bottom of the screen is an additional window which the student can use as a scratchpad and the designers can use to communicate with the underlying Lisp process. The menu bar contains "help" and "quit" entries which the student can use at any time to obtain information about how to use the interface, quit working on the current problem, or terminate the tutoring session.

Each session is logged to disk. For development purposes it is convenient to be able to play back a given session from the log without having to enter each of the student's responses in sequence. The logs are also useful as a source of data for analyzing human-computer conversations.

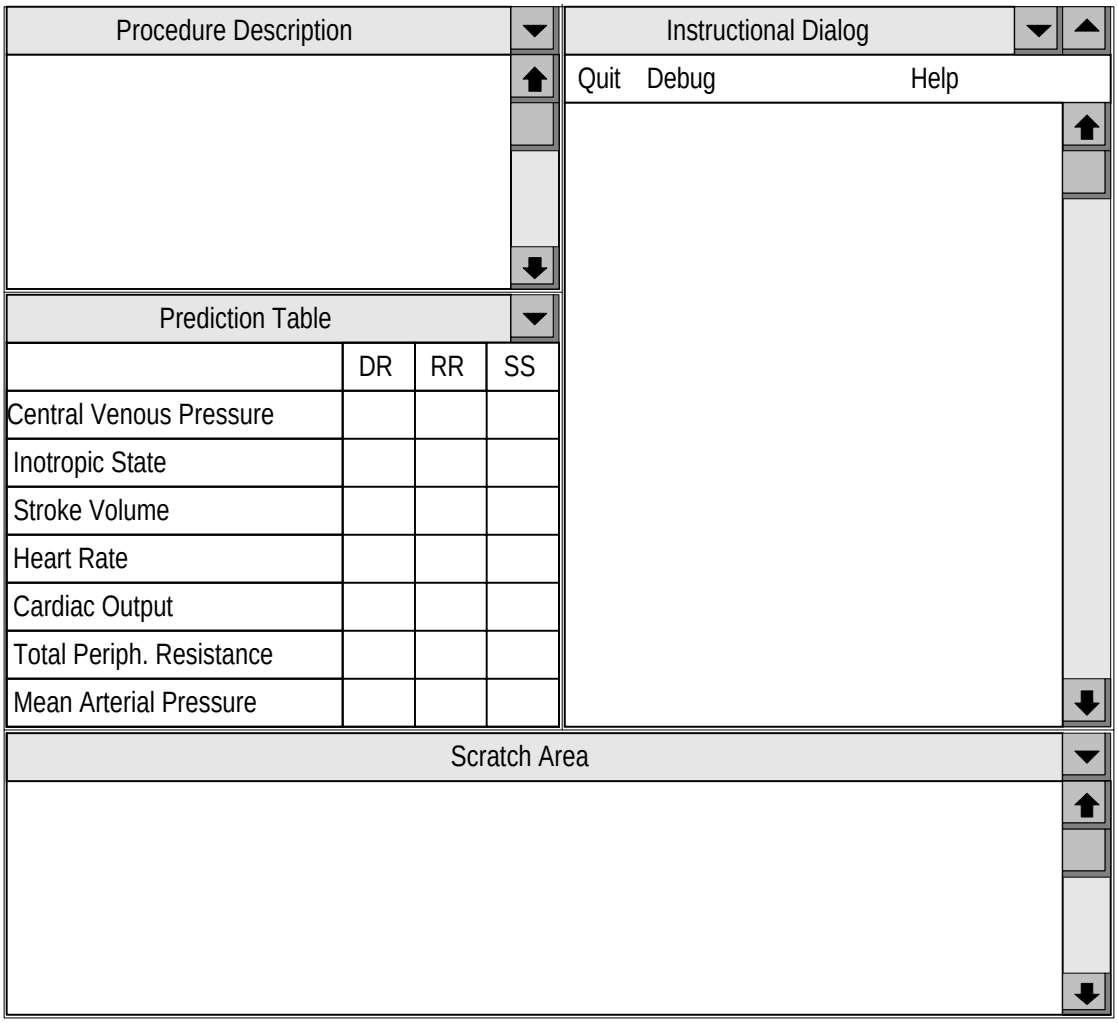

Figure 3.1: CIRCSIM-Tutor screen interface

3.3.2 Protocols for interleaving predictions and dialogue

After the student finishes each phase, the tutor and the student conduct a dialogue to give the tutor an opportunity to correct the student's misconceptions. In the protocol currently being used both for live tutoring sessions and for CIRCSIM-Tutor, called protocol 3 by Khuwaja et al. [1995], the student solves the problem in a largely uninterrupted fashion. At the beginning of the procedure, the student is required to

correctly predict the primary variable and its direction. After that, the student works the problem one stage at a time, making the predictions in any sequence. After each stage is complete, the tutor conducts a dialogue with the student where each incorrect variable is discussed.

The rationale for protocol 3 is that with the full set of predictions for a stage available, the tutor can have a better idea about the student's problems. Previous protocols interrupted the student after each wrong answer in line with more traditional pedagogical beliefs. On the other hand, CIRCSIM, a traditional CAI program described in Section 3.1.3, lets students work all three stages before intervening. A side effect of protocol 3 is that the tutor cannot require the student to predict the variables in a logical sequence, as there is no way to intervene during the prediction phase, although tutors do talk to the student about logical solution sequences.

In CIRCSIM-Tutor v. 3, the protocol is determined by a high-level schema. As a result, although the initial implementation uses the protocol described above, one could switch to a different protocol without changing any code.

Chapter 4
A model of instructional discourse for cardiovascular physiology

A good question is never answered. It is not a bolt to be tightened into place but a seed to be planted and to bear more seed toward the hope of greening the landscape of idea.

—John Ciardi

This chapter introduces a model for a tutoring dialogue based on the structure and discourse mechanisms found in naturalistic[1] data and suitable for implementation by an automated text planner. We analyze previously collected human-to-human tutoring sessions from a text planning perspective. We characterize the aspects of the tutor's productions which are the most significant to model in a computer-based tutoring system, including both content-based and content-independent patterns at sizes ranging from the conversation as a whole down to single turns. We illustrate patterns from the three aspects of each turn—responding to the student's last utterance, advancing the tutoring plan, and obtaining feedback from the student. Finally, we identify aspects of human tutoring which are not susceptible to current techniques in artificial intelligence.

Previous studies of the CIRCSIM-Tutor transcripts have concentrated on the syntactic structure of our human tutors' productions [Seu et al. 1991] or their pedagogical goals as indicated by surface structure [Hume 1995; Hume et al. 1993, 1995, 1996; Evens et al. 1993; Spitkovsky 1992]. Our analysis is oriented toward modeling the deep structure

[1] Although there were no constraints on the language to be used, the dialogues were collected in an experimental setting, not in everyday life.

of the human tutors' productions in order to generate similar ones.

4.1 Developing a model from naturalistic data

4.1.1 The keyboard-to-keyboard data collection sessions

Over the last four years, the CIRCSIM-Tutor project has collected a large number of transcripts of human-to-human tutoring sessions, including both face-to-face and keyboard-to-keyboard sessions. The corpus for this study consisted of 47 keyboard-to-keyboard transcripts totaling about 5000 turns[2] of dialogue, or about 75 hours of written dialogue. Our domain experts, who are experienced professors of physiology, served as tutors. The students were volunteer medical students who were paid a nominal fee for their participation. They are also told that they might learn something from the experiment: the slogan for recruiting volunteers is "earn while you learn." The sessions are conducted each year after the students have been introduced to the basic concepts in class but before they have completed the lab experiments, small group discussion sections and other activities which will complete their study of the topic. At this point, after the students have studied the basic material but before they have internalized it, they come closest to the ideal student for whom CIRCSIM-Tutor is being developed.

Because CIRCSIM-Tutor communicates with the student using written language—it displays text and the student types responses—the tutoring sessions in the corpus were conducted in the same way. In this manner we eliminate the expected differences between oral and written dialogues. A major difference between face-to-face and keyboard-to-keyboard sessions is the restricted availability of pragmatic cues such as intonation and gestures in the latter. Additionally, at present our experimental setup

[2] A *turn* is the connected speech of one speaker.

precludes the possibility of using graphics. Thus we need to find out how both student and tutor use language to make up for the missing possibility of communication via non-verbal channels. As a side note, collecting the data in written form eliminates the necessity for expensive and error-prone transcription, and aids in input analysis by providing data on misspellings and abbreviations invented by students.

Student and tutor are located in separate rooms so that they can communicate only via the terminal. The conversation is managed by the Computer Dialogue System (CDS) developed by Li et al. [1992] for this purpose. CDS uses a teletype-like interface which permits each party to see what is being typed by the other. CDS also permits interrupts from either party. In each tutoring session, the tutor helps the student work through one or two problems in cardiovascular physiology, with approximately one hour allocated to each problem. They use the same protocol as v. 3 of CIRCSIM-Tutor will eventually use, described in Section 3.3.2.

4.1.2 From tutoring session to transcript

The tutoring sessions are logged to disk. A suite of programs written by the CIRCSIM-Tutor group are used to format conversations and strip off personal identifying information, which is stored separately and used only for demographic purposes. Since it does no parsing, the numbering program cannot always correctly determine sentence boundaries, but can only guess based on punctuation. The resulting errors in sentence numbering are corrected by hand.

The transcripts have been stored in their original format so that future researchers will be able to study phenomena of their choosing. I designed an annotation subsystem which allows individuals to annotate interesting phenomena in the transcripts without interfering with other users' annotations. The annotation system permits us to produce

versions of the transcripts with normalized spelling and various types of formatting in order to improve the visibility of high-level phenomena such as pedagogical and discourse goals. Using a spelling-corrected version as input to our KWIC index programs would result in approximately a one-third increase in the number of useful entries retrieved.

4.1.3 Simplifying human discourse

One of the reasons human speech is so fascinating to study—and so difficult to model—is that there are multiple, overlapping meanings in even the shortest conversation. Human beings have been arguing for thousands of years about the intentions behind statements in Plato's dialogues. Although many statements can be considered as satisfying multiple goals, we will show that assigning a single meaning per statement at each level of planning, as a conventional planner does, is sufficient to model the dialogues of our expert tutors.

Since the model describes the deep structure of the dialogues, the mechanisms in this chapter do not necessary have one-to-one analogues in the surface structure, although many do. Thus these mechanisms are not necessarily the same as the discourse patterns identified by previous CIRCSIM-Tutor researchers through analysis of the surface structure of the transcripts. In Section 4.11, we will discuss how to implement previously described phenomena using the model described here.

We do not expect the human tutors to necessarily agree with or even be aware of the discourse mechanisms we use to model their behavior. For most people, discourse processing usually happens below the surface of consciousness except in writing or speech courses. Additionally, these discourse mechanisms have been chosen as a suitable model for an automated planner. Since human beings are capable of much more complex

reasoning, there is no need to persuade our expert users to see their speech the same way we do.

4.1.4 Typographical conventions used in this chapter

Excerpts from the transcripts are identified by transcript and turn number. The excerpts are quoted verbatim with the following exceptions: spelling has been corrected, punctuation and capitalization have been normalized, and abbreviations such as "I" or "+" for "increase" have been spelled out as forms of the verbs *increase*, *decrease* and *did not change*. I have consistently used abbreviations, such as CVP and IS, for core variables and spelled out the names of deeper-level variables. Omissions are indicated by ellipses and added material by square brackets. For simplicity, only the relevant section of each turn is quoted, frequently leading to the omission of the tutor's acknowledgment of the student's previous response and the tutor's closing request of the student. Where an additional explanation is necessary, it has been italicized and enclosed in parentheses. Arrows have been used to mark relevant text in long excerpts.

The schemata are described in pseudo-code. Since one goal of this dissertation is to provide a feasible architecture for CIRCSIM-Tutor v. 3, we have tried to demonstrate the range of possible schemata. We have also tried to identify a hierarchical set of schemata which provide the major functions necessary to make CIRCSIM-Tutor a useful resource for medical students. We have not tried to give exact prerequisites for the schemata shown, and we have freely used notation from extended BNF such as *if*, *repeat*, and *optional* where it simplifies the exposition. This notation may be implemented by the planner in a number of different ways.

Whether a text excerpt can be generated by a given schema depends not only on the schema but on the grammar and lexicon available and the realization rules used by the turn

planner to convert each set of semantic forms to text. We have tried to use only examples which could realistically be generated by the schemata given, but there is bound to be some disagreement. For some of the examples, to generate the exact text shown would require additional semantic forms, for example to generate a discourse marker or an adverb.

4.2 Principal aspects of the model

4.2.1 Modeling tutorial planning

The following list summarizes the major elements of the model. In the following sections we will describe each of them in more detail. Unless otherwise mentioned, each of these items applies both to the human tutors and to dialogues generated by CIRCSIM-Tutor.

- The tutor maintains a hierarchy of goals, visible via the hierarchical structure of the resulting conversation, while at the same time carrying on a conversation with an agent whose responses can't be predicted in advance.
- The text contains a segment for each of the three physiological stages. Within each stage, the text is divided into segments, one for each incorrect core variable. The variables are discussed in a partially ordered sequence which corresponds to the solution trace of the problem.

The fact that the basic building block of the model is the correction of one variable does not mean that that correcting variables is the human tutors' ultimate goal. In fact, the tutors' underlying goal is to teach the student several levels of information about solving problems in cardiovascular physiology, and their own model of the process concentrates on incorrect relationships between the variables rather than on incorrect values of individual variables. However, the simpler model proposed here, that tutors achieve their goals by tutoring one variable at a time, is sufficient for the purpose of text generation and

leads to simpler planning operators.

Both the human tutors and any mechanized system must steer a balance between teaching principles too directly, which does not help the student learn to apply them in context, and concentrating too much on the problem at hand, which could lead to the student memorizing the answers without understanding the domain reasoning.

- The text for each variable is divided into attempts to tutor it. The tutor makes attempts to teach a variable until the student has stated its correct value. The resulting hierarchical structure can be depicted as follows:

    Solve a problem
        Solve first phase
            Get student's predictions
            Conduct dialogue
                Correct first variable
                    First attempt
                    Second attempt
                    ...
                Correct next variable
                ...

- The tutor has a variety of methods for teaching the value of each variable, including single-turn and multi-turn methods. Methods may be nested, and one or more methods are available for tutoring each lower-level concept. If the student doesn't give the expected answer to a question, whether it is a question terminating a method or a question in an intermediate step, the tutor has three alternatives:

    a) Respond to the student, then continue with the method. The response may include an acknowledgment of the student's statement and/or a content-based reply. If necessary, the tutor can give the student the answer.

    b) Respond to the student, then retry the current goal with a new instantiation.

    c) Respond to the student, then back up one or more levels and try a new method. If the tutor backs out of a multi-turn method without

completing it, the remaining steps of the method are dropped. A cue phrase like "let's try again" is sometimes used to delimit attempts.

From a technical point of view, b) and c) are identical; in b) the tutor backs up zero levels.

- Semantic forms may be combined before text is uttered. This combination may involve combining clauses into one sentence, realizing a semantic form as a prepositional phrase or dependent clause, or even the realization of a semantic form through the use of a specific lexical item.

4.2.2 Modeling dialogue handling

The tutor must maintain the back-and-forth structure of a two-party conversation as well as carry out the hierarchical goals derived from the tutorial plan. In order to describe the tutor's possible response to the student, we must first classify the student's potential inputs. The following classification is sufficient for our needs:

- Positive response
    - Correct answer (direct or hedged[3])
    - Linguistically close but not exact answer
        Example: student identifies the correct variable but refers to it using terminology from the deeper-level concept map
    - A step toward the correct answer
        Example: student names a variable on the correct path in the concept map but not the desired variable

- Negative response
    - Wrong answer (direct or hedged)

- Student initiative
    - Question or statement which does not answer the question at the top of the agenda (i.e. the question at the end of the tutor's last turn)

[3] A *hedge* is text which indicates uncertainty, such as *I think*, *maybe*, or a question mark at the end of an otherwise declarative response.

- Multiple responses
  Any combination of the above, such as the following:
  - Correct and incorrect responses in the same turn
  - Positive/negative response plus student initiative
  - Correct response with incorrect reason

In the typical dialogue pattern, predicted by the Conversation Analysis school [Sinclair & Coulthard 1975; Stenström 1994] and observed in our transcripts, every turn has the following basic structure:

Turn:
  Response to student's previous statement (optional)
  New material (optional)
  Question or request for the student

The question is part of either the response or the new material. Since each part of the turn must be contiguous, the question can only belong to the response if there is no new material. The response itself can be divided into two sections as shown below. The question can only be part of the content-oriented section.

Response:
  Acknowledgment of student's statement (optional)
  Content-oriented reply (optional)

An acknowledgment may take one of two forms, or a combination of the two:

- A particle such as *yes* or *no*
- A more extensive statement such as *you're right*

We define *positive* and *negative* acknowledgments and content-oriented replies as those which could be used to respond to positive and negative utterances on the part of the student, as defined in the previous section. This definition accords with the usual use of the terms *positive* and *negative acknowledgments*. *Mixed acknowledgments* are used to respond to a multiple response, again as defined in the previous section.

The most common forms for negative content-oriented replies include the following:

- A statement denying the student's statement or a related fact
- A rebuttal of the student's statement or a related fact
- A statement supporting the opposite of the student's statement

An explicit negation of the student's statement is an emphatic way to point out a wrong answer. Sometimes it is more useful to negate a part of the student's statement or a logical prerequisite to it. A rebuttal is used when the tutor wants to share some reasoning about the student's error, not just the facts. A statement supporting the opposite conclusion can be a useful hint toward the right answer. Replies to the student can invoke multi-turn plans, but in general only rebuttals are complex enough to require one.

The most common forms for positive content-oriented replies include the following:

- A restatement of the student's statement, possibly in more precise language
- A statement supporting the student's statement

These replies are used in response to a correct answer or an answer which is basically correct but where the tutor wants to improve the student's language. When the student says something which is on the path to a correct answer but which needs further dialogue, the tutor's most common response is to switch to a new schema which refers to a lower-level concept map. A positive acknowledgment may be issued but there is usually no need to issue a content-oriented reply because the new schema will respond to the content. This case will be described in Section 4.10.

After replying to the student's statement, the tutor may continue with new material from the teaching plan. The turn ends when the tutor must request information from the student. The request is most commonly phrased as a question, but it can also be expressed as an imperative. It can be framed in direct or indirect language ("Can you tell me…").

In the transcripts the closing question or request is sometimes left implicit. For

example, if the tutor states a fact which hints at a solution to a problem, the student will probably try to solve the problem without an explicit request [Schegloff & Sacks 1973; Grice 1975]. For a number of reasons, discussed in detail in Section 5.3.7, the mechanized tutor always ends each turn with an explicit question or request.

4.2.3 Examples showing multiple attempts and turns

The first example shows an incorrect prediction by the student followed by two attempts at correction. Each attempt ends with the tutor asking for the value of the variable to see if the student understands yet. After the second attempt, the student gives the correct answer.

(1) S: CC increases.
T: Yes, that's the effect of increased sympathetic stimulation on the myocardium. However, what happens to CC in the DR period?
S: CC increases.
T: Reminder, the DR occurs before there are any reflex (neural) changes. What happens to CC in the DR?
S: CC remains constant.

(K3:40–44)

The tutor's first turn includes a response followed by new material. The response consists of an acknowledgment plus a statement supporting the student's utterance. The new material includes the concept "we're in DR now," expressed by the phrase *in the DR period*, and a question about the value of the variable. The tutor's second turn consists entirely of new content.

It is sometimes unclear whether to classify a statement like the reminder in the tutor's second turn as a reply or as new content. The *let's try again* test sometimes provides a useful heuristic in such cases. Since the cue phrase *let's try again* is used to start a new attempt, it separates the reply from the new content. In other cases, our

research has not yet led to a principled way to classify such statements. We expect that attempting to generate coherent text with CIRCSIM-Tutor may clarify this issue.

The second example demonstrates the use of a multi-turn plan. The tutor wants to point out a contradiction based on the fact that MAP = CO * TPR.

(2) T: … What are the determinants of MAP?
S: CO and TPR.
T: Correct. And you have predicted CO increases and TPR increases. So how can you say MAP decreases?
(K32:226–228)

If the student had not "played along" with the tutor by correctly answering the first question, the tutor would have been unable to continue as planned. In fact, that's exactly what happens in the next example:

(3) T: Now look at your predictions: MAP decreases, TPR increases, CO doesn't change. Is that possible?
S: Yes.
T: Let's try again. MAP = CO * TPR. CO doesn't change. TPR increases…
(K33:122–124)

In this case, the tutor recovers by trying the same plan again but wording it differently. The cue phrase "let's try again" indicates a new attempt.

Schema switching will be discussed in Sections 4.6.6 and 4.10.1.

4.2.4 Realizing a semantic form in multiple ways

In addition to using a variety of argumentative forms to tutor each concept, the tutor can also express the communicative acts which comprise the forms in more than one way. The sentences below show several locutions which the human tutors use to start the dialogue. Although each of these serves the same purpose, they have slightly different semantic values. From a pedagogical point of view, the primary value of having multiple

realizations of a concept may be to encourage the student to read and understand the material, rather than to respond by rote. Additionally, the use of a variety of ways to express a concept exposes the student to more of the medical sublanguage which our domain experts want them to acquire. Finally, one of the realizations may help the student to recall existing knowledge. From a technical point of view, having multiple verbs and syntactic configurations available permits us to choose a different realization when the tutor needs to ask the same question twice. Depending on the rest of the turn, multiple potential realizations may be necessary in order to have one which fits into the turn being planned.

The following examples show alternative realizations for *introduce-stage*.

(4) T: … Now let's review some of your predictions…
(K32:46)

(5) T: … Let's take a look at some of your predictions…
(K10:29)

(6) T: … let's talk about your predictions…
(K27:50)

(7) T: … There are some errors here. Let's start with this issue…
(K14:31)

The following examples show a set of alternatives for a key component of *move-forward*.

(8) T: … And what happened to RAP?
(K28:8)

(9) T: … What effect would this have on RAP?
(K32:62)

(10) T: … what must happen to SV?
(K28:10)

(11) T: … How will that affect SV?
(K30:74)

Any of the following alternatives can be used to return to the top level of the concept map.

(12) T: … What variable in our list reflects this?
(K28:5)

(13) T: … What variable has the same value as CVP?
(K6:35)

(14) T: … What parameter here reflects filling of the left ventricle?
(K27:66)

Finally, the following examples show a variety of ways to inquire about the determinants of a variable.

(15) T: … But let's consider … the things that determine SV. Can you list these for me?
(K33:74)

(16) T: What are the two parameters that determine CO?
(K13:49)

(17) T: … What parameter determines RAP?
(K25:48)

(18) T: … What are the determinants of SV?
(K15:59)

(19) T: … Can you write the equation relating MAP, CO, and TPR?
(K24:96)

Section 5.2.3 gives examples of the independence of the generated surface structure and the underlying semantic form.

4.2.5 Naming convention for schemata

Although names are arbitrary, we have attempted to name schemata consistently in order to show patterns where they exist and reduce the cognitive load on the reader. In this chapter the schemata are named from the tutor's point of view; the first word of each schema name is a verb describing the tutor's action. In Chapter 5 the plan operators are named from an objective point of view to emphasize the point that some conversational

objectives can be achieved by either party. As CIRCSIM-Tutor moves closer to the goal of cooperative communication, the use of an objective point of view becomes more important.

The following list includes the most common verbs used in the schema names:

Forms relating to dialogue sections:

Process- to get predictions for a section of the problem, then conduct a dialogue
Start- to begin a section of the dialogue (first time or retry)
Introduce- to begin a section of the dialogue (first time)
Attempt- to converse with the student about a topic. Because we cannot predict the student's response in advance, an attempt may not be successful.
Correct- to converse with the student about a topic, leading to a correct answer about a question
Conclude- to end a section of the dialogue

Forms relating to domain knowledge:

Teach-info to communicate some information, either via *convey* or *elicit*
Convey-info to give the student information about a topic
Elicit-info to request information from the student
Get-<variable-name> abbreviation for eliciting the value of *variable-name*
Ensure-student-knows to teach the specified information unless it has been conveyed earlier in the conversation

From a philosophical point of view, schemata names are only a mnemonic aid to the reader [McDermott 1981]. We conventionally assume, as does Moore [1995], that students know something when they have been told. From a pedagogical point of view, several responses over a period of time may be required to really ensure that a student knows something. Even in that case, we can never know whether the student will retain the knowledge over the long term.

4.3 Discourse mechanisms used in high-level planning

4.3.1 Top level: interleaving data acquisition and correction

The top-level rules structure the task according to a protocol selected by our domain experts. In the currently preferred protocol, as described in Section 3.3.2, the student solves each stage uninterruptedly before discussing it with the tutor. This protocol is used in the transcripts beginning with K30, and can be realized with the following schemata:

Correct-problem:
    Process-primary-variable
    Process-stage (DR)
    Process-stage (RR)
    Process-stage (SS)

Process-primary-variable:
    Get-predictions (DR, ?primary-variable)
    Correct-variable (?primary-variable)

Process-stage (?stage):
    Get-predictions (?stage, all-remaining)
    Correct-stage (?stage)

The form *get-predictions* is a primitive which obtains the student's initial predictions for the variables using the graphical interface described in Section 3.3.1.

4.3.2 Introducing a stage

In this section we give some potential realizations for the schema *introduce-stage*, which performs the pedagogical function of introducing the stage to be discussed next. Although the pedagogical function of introducing a topic may be implemented, as least in part, by the discourse function of introducing a topic, the two are different in other contexts.

(4) T: … Now let's review some of your predictions…
(K32:46)

(5) T: … Let's take a look at some of your predictions…
(K10:29)

(6) T: … let's talk about your predictions…
(K27:50)

(7) T: … There are some errors here. Let's start with this issue…
(K14:31)

The examples above show the result of realizing *introduce-stage* in a sentence by itself. The following turns show two different ways to implement *introduce-stage* plus the beginning of the first variable in *correct-variables*.

(20) T: … That completes your DR predictions. Most of them are correct. However, I want to pursue IS with you. Can you tell me what you think that IS means?
(K47:56)

(21) T: Let's take these one at a time, starting with CC…
(K5:29)

4.3.3 Correcting a stage

The following schemata can be used to correct a stage:

Correct-stage:
- Introduce-stage (optional)
- Correct-variables
- Inquire-about-correct-variables (optional)
- Conclude-stage (optional)

Correct-variables:
- Correct-variable*
- Conclude-correction-phase (optional)

The semantic form *inquire-about-correct-variables* permits the tutor to check the student's understanding even if the student has predicted all the variables correctly. This item, which is optional, is not discussed further here because the open-ended nature of the

questions used (see the next example), make it difficult to implement directly.

4.3.4 Concluding a stage

Here is a turn which contains *conclude-correction-phase* plus the beginning of the first variable from *inquire-about-correct-variables*.

(22) T: … All of your other predictions were correct. However, I'd like to know how you arrived at some of them. For instance, how did you come up with CVP decreasing?

(K47:62)

Here is an example of *conclude-stage*. This semantic form is often omitted, as the tutors usually just move to the next stage.

(23) T: You got it. And you have finished the DR…

(K5:27)

4.4 Discourse mechanisms used in correcting a variable

4.4.1 Second level: division into variables

As described in Section 3.3.2, the tutor tutors the student on each incorrect variable in a sequence based on a solution trace of the problem. Rules for determining the solution trace are given in Section 2.4. For example, in the DR stage, the variables are usually discussed in the following order:

- Neural variables
- Variables on the shortest path to MAP
- Variables on the "secondary path"

The fact that the dialogue is planned one variable at a time does not restrict the algorithms available to the pedagogical problem solver for choosing the sequence of variables to correct. For example, the pedagogical problem solver could use an algorithm involving

nested variables ("to correct V_1, correct V_2 first") to determine which variable to teach next. The effect of nested variables can be obtained without an explicit rule like the one above by making the correction of V_2 a prerequisite of the rule which corrects V_1.

In addition to nested pedagogical goals, other types of pedagogical planning algorithms could be implemented by changing only the pedagogical problem solver. For example, adding the ability to utilize the fact that the student has given the correct answer for a variable other than the one being discussed, a form of opportunistic planning, would not require any changes to the planner. The ability to drop a variable and return to it later, or the ability to compare cases, heavily used in the Socratic dialogues of Collins [Collins & Stevens 1982], could also be added without difficulty, but neither is expected to be useful since the solution path for most CIRCSIM-Tutor problems is close to being linearly ordered.

4.4.2 Introducing a variable to correct

The schema for correcting a variable looks like this:

Correct-variable:
- Start-new-variable (optional)
- Attempt-correction-section$^+$
- Conclude-variable (optional)

The semantic form *start-new-variable* can be omitted, or implemented in one of two ways.

- Introduce-variable
- Transition-to-next-variable[4]

The semantic form *introduce-variable* can be used at any time, but *transition-to-next-variable* is not valid for the first incorrect variable of a stage. Here is an example of *introduce-variable*:

[4] Since our naming convention requires that semantic forms start with a verb, this must be the ugly neologism "to transition."

(24) T: … You predicted that TPR would increase…
(K48:44)

The following two examples show *transition-to-next-variable* realized by itself.

(25) T: … Let's also look at your prediction for SV.
(K27:62)

(26) T: … Now let's get back to some other things…
(K5:41)

The following two examples show combinations of *transition-to-next-variable* with other semantic forms. In the first example, *transition-to-next-variable* is used to generate the particle "now", while the rest of the sentence is generated by a semantic form such as *elicit-info*. In the second example, the tutor starts a new variable after a long correction for the previous one. The phrase "while we're on the subject" is generated by *transition-to-next-variable* and the rest of the sentence is generated by *correct-additional-neural*.

(27) T: *(gives answer for RAP)*
Now, what will happen to SV?
(K2:32)

(28) S: *(gives answer for one of the neural variables)*
T: … While we're on the subject, what other variable is under neural control and how will it be affected in the DR?
(K5:33)

In the following example, *transition-to-next-variable* is implemented with a schema which asks the student for the name of the variable.

(29) T: … Let's come back to an earlier prediction of yours. You said (correctly) that CO increases. What variable would that affect?
(K6:27)

4.4.3 Transition between attempts to correct a variable

Sometimes the student still doesn't understand after the tutor's first attempt to

explain. Each attempt except the last ends in the student giving a wrong answer. Once the student gives a right answer, the correction of the variable is complete, and no more attempts are required.

The form *start-new-attempt* may be used to generate an introductory word, phrase, clause or sentence preceding the correction attempt:

Attempt-correction-section:
    Start-new-attempt (optional)
    Attempt-correction

Just as at the beginning of a new variable, there are two ways to introduce a new attempt at correction, one which is always valid and one which can only be used after the first attempt:

- Introduce-attempt
- Transition-to-next-attempt

Here are two examples of *transition-to-next-attempt*:

(30) T: … What will happen to RAP then?
S: *(gives unsatisfactory answer)*
→ T: Let me start again.
(tutor starts new explanation for value of RAP)
(K2:30–32)

(31) T: Can you now tell me what determines RAP?
S: Central venous pressure.
→ T: Let's try it this way…
(K14:37–39)

The following excerpt shows *start-new-attempt* used as part of a first attempt to correct a variable. This case is rare in our current protocol because it is usually subsumed by *start-new-variable*. In this example, the student introduces the variable, leaving the tutor free to introduce the first correction attempt without redundancy.

(32) T: … What variable would change next?
S: TPR.
T: In what direction do you think it would change?
S: TPR would increase.
→ T: Let's think about this. TPR is a neurally controlled variable. Simply changing the CO wouldn't affect its value, would it?
(K4:23–27)

4.4.4 Concluding an attempt: Asking for a new value for a variable

The tutor ends each attempt by determining whether the student now knows the correct value of the variable.

(33) T: … What will happen to RAP then?
(K2:30)

(34) T: Can you now tell me what determines RAP?
(K14:37)

(35) T: … So what about TPR?
(K10:31)

(36) T: So TPR?
(K10:37)

Since by definition, the tutor's turn ends when it is the student's turn to speak, the end of an attempt is always the end of a turn.

4.5 Responding to the student's turn

In this section we describe each type of response mentioned in Section 4.2.2.

4.5.1 Acknowledgments: Positive, negative and mixed

Positive acknowledgments are especially likely to be explicit when the student has correctly responded to an intermediate question in a multi-turn plan. In the following example, the student has just responded correctly to the first step of the *correct-neural* plan (Section 4.6.1). The tutor responds with a positive acknowledgment followed by new

material.

(37) T: … Can you tell me how TPR is controlled?
S: Autonomic nervous system.
→ T: Yes. And the predictions that you are making are for the period before any neural changes take place. So what about TPR?
(K10:29–31)

The following excerpt contains another example of a positive acknowledgment followed by new material. In addition, it shows a negative acknowledgment followed by some information supporting an answer other than one given by the student. The fact that these questions do not deal with the values of variables demonstrates that the tutorial mechanisms in this example are being used to satisfy subgoals below the core variable level.

(38) T: … When MAP goes up, what change occurs to the autonomic outflow to the arterioles?
S: The efferent outflow causes vasodilation of arterioles.
→ T: That's right. What nerves are affected and in what way?
S: Sympathetic cholinergic nerves.
→ T: No. They're not part of the baroreceptor reflex. The sympathetic adrenergic vasoconstrictor nerves are…
(K1:50–54)

In the following example, a negative acknowledgment is followed by new material. The new material is signaled by a cue phrase generated by the *start-new-attempt* pattern.

(39) *(after a question and wrong answer, but before the correction)*
T: Not really. Try to look at it this way. The right atrium is like the end of the venous system…
(K30:66)

If the tutor asks for more than one value, such as requesting multiple determinants for a variable, the student may get one right and one wrong. In such a case, the tutor can usually ignore the right answer and deal with the wrong one using any of the methods for

correcting a single wrong answer. Before doing so, however, the tutor must acknowledge the student's response. Although the method used for this purpose is sometimes called "partial acknowledgment," the term "mixed acknowledgment" is more accurate. The student's entire answer is acknowledged, but the tutor only accepts part of it.

In the following example, a mixed acknowledgment is followed by new material designed to teach the student about the variable which was missed.

(40) T: What's your first prediction?
S: TPR, HR, CC all increase simultaneously.
→ T: That's pretty good except for HR. Remember in this case this guy's HR is solely determined by his broken artificial pacemaker.
(K18:36–38)

In the next example, the mixed acknowledgment is followed by a restatement of the correct answer in different words. The new material in this excerpt consists of a follow-up question designed to get the schema back on track.

(41) T: … What are the main determinants of SV?
S: Atrial filling and HR.
→ T: I agree with the first of these. Filling pressure is one. What's the other main determinant of SV?
(cf. K3:33–35)

A final example of mixed acknowledgment makes clear that a mixed acknowledgment always deals with more than one value.

(42) T: … Can you tell me what parameter in the table represents preload?
S: RAP or TPR.
→ T: RAP yes! TPR no. TPR determines MAP, which is afterload…
(K31:72–74)

4.5.2 Negative content-oriented responses

The transcripts contain two common types of negative content-oriented responses:

- A statement denying the student's statement or a related fact
- A statement supporting the opposite of the student's statement

There is not always a clear dividing line between these categories. Similarly, when the tutor gives a hint about a different answer, it is not always clear whether to classify it as an aspect of a negative reply or as new content, although the "let's try again" test is often useful. Sometimes a statement by the tutor appears to be a reply when discourse issues are considered but appears to be new content when looked at from the point of view of pedagogy. In general, these ambiguities do not cause problems because they do not affect the text which can be generated.

When the tutor issues a denial, the item most commonly denied is a logical precursor to the student's statement. For example, returning to example (38), the fact that the sympathetic cholinergic nerves are not part of the baroreceptor reflex means that they cannot be affected as the student has suggested. The tutor follows up the reply by giving the correct answer.

(38) T: … When MAP goes up, what change occurs to the autonomic outflow to the arterioles?
S: The efferent outflow causes vasodilation of arterioles.
T: That's right. What nerves are affected and in what way?
S: Sympathetic cholinergic nerves.
→ T: No. They're not part of the baroreceptor reflex. The sympathetic adrenergic vasoconstrictor nerves are…

(K1:50–54)

Although the tutor may use words or phrases from the student's statement, the tutor generally does not deny a statement by negating the student's exact statement. Sometimes the tutor denies the student's rationale for an answer but not the answer itself.

In example (42), the statement "TPR determines MAP, which is afterload" is not a direct denial of the student's idea that TPR represents preload. However, it does support the idea that preload must be represented by a different variable.

(42) T: … Can you tell me what parameter in the table represents preload?
S: RAP or TPR.
→ T: RAP yes! TPR no. TPR determines MAP, which is afterload…
(K31:72–74)

When a statement supporting the opposite answer is issued, the supporting material is usually a logical prerequisite of the correct answer. When combined with a positive acknowledgment, this type of reply is designed to reinforce the student's answer by enriching the student's knowledge. When combined with a negative acknowledgment, the goal is to encourage the student to rethink the answer. In the following example, instead of telling the student directly that HR will not be affected, the tutor gives a prerequisite for deducing that fact.

(43) T: OK, but which variable will be affected by the [alpha] agonist directly?
S: HR increases.
→ T: No. Alpha receptors are not present on the SA node.
(K31:16–18)

This example has an ambiguous status. From the point of view of discourse analysis, "alpha receptors are not present on the SA node" is a response to the student, but from a pedagogical point of view, it is a hint toward the correct answer. The implication is that hints can be found in the response portion of the turn as well as in the new material. The following example is similar.

(44) S: SV does not change. It will not be affected in the DR.
T: Not true. SV is only partially under neural control via CC, one of its input determinants…
(K5:34–35)

The following example contains a student initiative describing a misconception and the tutor's response. The tutor corrects the misconception obliquely. This example shows that a "negative" response to a student's statement need not contain any overtly negative linguistic elements.

(45) S: SV stays the same.
T: When MAP increases, it's harder for the ventricle to pump blood. So what would that do to SV?
S: SV would decrease. I was thinking that CC staying the same would not allow more blood to be pumped out of the ventricle.
→ T: You need to take all of the determinants into consideration together…
(K31:79–82)

Providing a specific rejoinder to a student's incorrect response can be difficult for two reasons. First, in order to give a specific response we must understand the student's statement. Incorrect answers do not necessarily map well onto the domain model. Second, it is not possible to have schemata available for responding to every possible wrong answer, although it is desirable to have as many as possible available in the plan library. The following example shows a specific rejoinder.

(46) T: Now how about TPR?
S: I'm thinking that it will increase very briefly but immediately decrease so as to adjust back to normal the CO.
T: By what mechanism will it increase?
S: If you increase pressure will you momentarily increase resistance.
→ T: No. You may be thinking of autoregulation. That's slow. Remember that we're dealing with the short period before you get a reflex response. Is this what you had in mind?
(K12:31–35)

In the following example, the specific rejoinder contains the tutor's attempt to correct one of the most common student misconceptions. Although one of the tutor's goals is to make sure that the student knows the difference between contractility and the Frank-Starling

effect, the tutor generally only has occasion to mention it if the student gets an appropriate variable wrong.

(47) T: Let me try this another way. How is the inotropic state of the heart *(i.e. CC)* altered physiologically?
S: By the force-length relationship?
T: No! You are confusing the Frank-Starling effect (increased filling gives greater output) and contractility. CC is the force of contraction at any given filling…
(K26:52–54)

4.5.3 Pseudo-diagnostic questions

In the following example, a denial is implemented as a question.

(48) T: There are some errors here. Let's start with this issue. What parameter determines RAP?
S: MAP.
→ T: Can you explain the mechanism by which MAP determines RAP?
(cf. K14:31–33)

We call this pattern the "pseudo-diagnostic question" because it has the same form as a true diagnostic question but is not intended to elicit an explanation, as the tutor knows that MAP does not determine RAP.

Occasionally, as in the following example, a student finds a way to respond to a pseudo-diagnostic question other than by saying "no." This case is treated just as any other unexpected response, i.e. a new schema is chosen. Although v. 3 of CIRCSIM-Tutor does not ask open-ended questions, the fact that a true diagnostic question would have the same form as the pseudo-diagnostic question would not cause any problem, as we would choose a schema based on the student's response in any case.

(49) T: … Now let's get back to CO. You said that it would increase. What variable would that affect?
S: SV.

→ T: How would it affect SV?
S: SV increases.
→ T: In what way can CO affect SV?
S: CO = HR * SV, so an increased CO would mean an increased SV.
T: In this case, remember, CO went up because HR increased. So, an increase in CO does not mean that SV went up. Can you tell me what the main determinants of SV are?

(K4:35–41)

4.5.4 Pointing out a contradiction

One way of pointing out to the student that an answer is wrong is to demonstrate that the answer leads to a contradiction. In the *show-contradiction* method, we take one or more statements of the student and derive a contradiction from them.

In the following example, the tutor shows that the student is contradicting rule *Prop-2u* of Section 2.4.2 about the propagation of values.

(50) T: … RAP and CC determine SV. You predicted that CC would be unchanged and that RAP increased. How can SV be unchanged?

(K27:68)

The following example shows a contradiction to rule *Prop-2e* of the same section. Remember that MAP = CO * TPR.

(51) T: … What are the determinants of MAP?
S: CO and TPR.
→ T: Correct. And you have predicted CO increases and TPR increases. So how can you say MAP decreases?

(K32:226–228)

The following text shows two consecutive attempts to correct a variable, each instantiated with *show-contradiction*:

(52) → T: Now look at your predictions: MAP decreases, TPR increases, CO doesn't change. Is that possible?
S: Yes.

→ T: Let's try again. MAP = CO * TPR. CO doesn't change. TPR increases. MAP should increase by that logic. But you said the reflex doesn't completely correct MAP. *(i.e. MAP decreases)* So?

(K33:122–124)

Conversely, the following excerpt is syntactically consistent with the output of *show-contradiction*, but in fact shows the tutor asking a real question.

(53) → T: So we have HR going down, SV going up and CO going down. How is this possible?
S: HR is down more than SV is up.
T: Very good.

(K27:72–74)

As in the case of the pseudo-diagnostic question, if the student tries to answer a *show-contradiction*, the tutor can choose a new schema to handle the situation.

4.5.5 Positive content-oriented responses

The transcripts contain two common types of positive content-oriented responses which can be used to respond to correct answers or correct answers where the tutor wants to improve on the student's language.

- A restatement of the student's statement, usually in more precise language
- A statement supporting the student's statement

The following excerpts contain examples of restating the student's statement as a way of reinforcing an important point.

(54) S: CC does not change because there is no change in the sympathetic innervation on the heart with the change in the pacemaker. TPR does not change because of the same reason.
T: Another way of saying this is that both CC and TPR are determined by the reflex and the reflex hasn't happened yet in DR

(K24:43–44)

(55) T: … Now, when CO falls, how does that affect RAP?
S: RAP increases.

T: Super. They are inversely related when CO changes before RAP does…
(K31:84–86)

The following excerpt shows an example of supporting the student's statement. Perhaps the tutor provided explicit positive feedback in this turn because of the negative feedback provided in the tutor's previous turn.

(56) T: OK. Then predict the first variable that would be affected by injecting the alpha agonist.
S: CC.
T: The myocardium only contains beta receptors and hence can't respond to an alpha agonist.
S: TPR.
→ T: Sure. The vascular smooth muscle contains alpha receptors…
(K32:18–22)

4.5.6 Responses to student initiatives

When the student changes the topic of conversation, either instead of or in addition to replying to the tutor's question, the tutor has several options for responding:

- Put the student's request on the agenda above current plan (i.e. respond to student's statement, then return to plan)
- Put the student's request on the agenda instead of current plan (i.e. switch to new plan)
- Put the student's request elsewhere on the agenda
- Acknowledge student's input without responding
- Ignore student's input

In effect, the tutor's options for responding to a student initiative are similar to the tutor's options for responding to any unexpected input, such as a wrong answer.

If the student responded to the tutor's question in addition to uttering the initiative, the tutor must decide whether or not to respond to that aspect of the student's utterance. If the student's answer is incorrect, the tutor will generally need to respond to it unless the content of the reply has already been covered by the response to the student initiative.

In the following examples, the tutor replies to the student's statement, then returns to the tutoring plan in progress. Responding in this fashion is a sensible choice when the student initiative consists of a request for a definition, an explanation, or help, as in the first example. In the second example, the tutor responds to a student statement for which the truth value is can be determined. This type of student initiative is more difficult for CIRCSIM-Tutor because the input understander may need to expend considerable resources before it can determine whether such a statement is within its purview or not.

(57) T: … What would happen after that?
S: Before I answer, could you explain the difference between SV and CO?
T: SV is the amount of blood pumped by a ventricle in one beat. CO is the volume of blood pumped by a ventricle in one minute.
(K34:30–32)

(58) T: What's your first prediction?
S: TPR, HR, CC all increase simultaneously.
T: That's pretty good except for HR. Remember in this case this guy's HR is solely determined by his broken artificial pacemaker.
→ S: Wouldn't his other myocardial cells respond to sympathetic stimulation and couldn't they override his artificial pacemaker?
T: They might and then again they might not. We're assuming in this case that they don't. So what do you say about HR?
(K18:36–40)

A potential student initiative is the case where the student opportunistically responds to a higher-level goal on the agenda or gives a value for more than one variable. CIRCSIM-Tutor could use the information to shorten the dialogue, but we have not planned to do so because our domain experts prefer to see each variable discussed.

In all of the cases above, the student's desire is incorporated into the tutor's agenda. Initiatives which would require maintaining two goal stacks, one for the tutor and one for the tutor's perception of the student's goals, as in a truly cooperative conversation, are outside the bounds of CIRCSIM-Tutor.

4.6 Correcting neural variables

This section describes the principal method used by our human tutors to correct the student's ideas about neural variables in the DR stage. In the following two sections, we look at equivalent methods for non-neural variables and for the primary variable. In cases where different rules are required for the three physiological stages, we show detailed rules only for the DR stage.

4.6.1 Schema for correcting the first neural variable

The following schema shows the method most frequently used by our tutors to correct the first neural variable which the student has predicted incorrectly.

Correct-neural (?v):
PQ: ?v is neural
?v is not the primary variable
Ensure-student-knows (has-mechanism(?v, neural))
Ensure-student-knows (not(is-active(nervous-system, DR)))
Ensure-student-knows (has-value(?v, DR, no-change))

The intended interpretation is that the tutor must ensure that the student knows the following three facts, which constitute a syllogism for teaching the value of a neural variable.

The variable is controlled by the nervous system.
The nervous system has no effect in the DR phase.
Therefore the value of the variable does not change in DR.

When it is desirable to represent the concept "therefore" explicitly, a semantic form is added for that purpose. For simplicity, we will not show this form in the examples.

The most common implementation is for the teacher to tutor the concepts involved and then ask the student for the value of the variable, i.e. as if the schema were written as follows:

Correct-neural (?v):
Teach-info (has-mechanism(?v, neural))
Teach-info (not(is-active(nervous-system, DR)))
Elicit-info (has-value(?v, DR, no-change))

The main difference between this schema and the previous one is that this schema requires some text to be generated for each semantic form, while *ensure-student-knows* can evaluate to *nil* if the tutor believes that the student already knows a fact.

Since *teach-info* can be satisfied by telling the student something (*convey-info*) or by eliciting the information from the student (*elicit-info*), the semantic form *ensure-student-knows* can be realized in three ways, as shown below.[5]

- *Convey-info*: Give the student some information
- *Elicit-info*: Ask the student for some information
- *Nil*: Say nothing

This choice of methods can be used to generate a wide variety of observed tutorial phenomena.

4.6.2 Instantiation as interactive explanation

The following set of choices occurs frequently in our transcripts:

Elicit-info: Ask student for the control mechanism for the variable
Convey-info: Inform student that nervous system did not activate yet
Elicit-info: Ask student for the correct value of the variable

This option can be used to generate examples such as the following:

(37) T: … Can you tell me how TPR is controlled?
S: Autonomic nervous system.
T: Yes. And the predictions that you are making are for the period before any neural changes take place. So what about TPR?
(K10:29–31)

[5] In Section 5.2.3, we show that the choice of semantic form does not determine the eventual surface syntax.

This text is an example of an *interactive explanation*, a generalization of Sanders' [1995] "directed line of explanation" (DLR). An interactive explanation conveys information to the student using at least one question in addition to the final request for data.

4.6.3 Instantiation as explanation

If the student finds interactive explanation too difficult, the tutor can try a less interactive option. A natural option is for the tutor to give an *explanation*, then ask a question to make sure the student understands. This choice could be described as follows:

Convey-info:	Inform student that variable is neurally controlled
Convey-info:	Inform student that nervous system did not activate yet
Elicit-info:	Ask student for the correct value of the variable

The following excerpt illustrates this alternative.

(59) T: … Remember that TPR is neurally controlled and that we're discussing what would happen in the DR period. That's before there are any reflex changes. Try again.

(K7:23)

The following example uses the same instantiation of the schema, but the semantic forms have been amalgamated into one sentence. The final question has been left in a semi-explicit state, something which is acceptable in a human-to-human conversation but not in CIRCSIM-Tutor.

(60) T: If CC is under neural control and we're talking about the period before any change in neural activity then CC—

(K11:57)

The following example is similar, except that the order of the first two semantic forms has been switched.

(61) T: Remember, we're dealing with the direct response period. That's before there are any reflex responses. TPR is a neurally controlled variable. Try again.

(K6:23)

Although CIRCSIM-Tutor always requests an explicit value for each variable, the human tutor is not so restricted. Thus the human tutor could instantiate all three semantic forms with *convey-info*, as follows:

Convey-info: Inform student that variable is neurally controlled
Convey-info: Inform student that nervous system did not activate yet
Convey-info: Inform student of the correct value of the variable

This set of implementation choices creates monologic explanations such as the following:

(62) T: … But more to the point, both TPR and CC change only when the reflex alters the activity in the autonomic nervous system. And since DR is before the reflex can act, both must be unchanged in DR…

(K13:57)

This type of explanation is relatively uncommon because our human tutors prefer to elicit the value of the variable even if the rest of the schema has been instantiated in a non-interactive form. In the following similar example, the human tutor follows up the explanation with a yes/no question. We do not need to generate this option because yes/no questions do not provide useful information to a mechanized tutor.

(63) T: HR and CC are unchanged because they are neurally determined variables and the nervous system isn't involved in DR. OK?

(K35:114)

4.6.4 Instantiation as hint

Another option is to instantiate one of the first two semantic forms as *nil*, resulting in a setup such as the following:

Nil:	(No text generated here)
Convey-info:	Inform student that nervous system did not activate yet
Elicit-info:	Ask student for the correct value of the variable

This option is used to generate a *hint* for the student. In particular, this instantiation generates a CI-hint[6] in the terminology of Hume et al. [1993]:

(64) T: … Remember that we're talking about what happens before there are any neural changes. Now what do you say about TPR?
(K11:51)

The following example is similar, except that the open-ended question is more suitable for a human tutor than for CIRCSIM-Tutor.

(65) T: … Now I would remind you that you are predicting what would happen before there are any reflex changes. Would you like to change any of your predictions?
(K31:56)

The following example combines a question-less hint with a pseudo-diagnostic question.

(66) T: But remember that we're dealing with the period before there can be any neural changes. How can CC go up if it's under neural control?
(K10:43)

4.6.5 Another schema for correcting the first neural variable

The following schema is an alternative to the *correct-neural* schema introduced in Section 4.6.1.

Correct-neural-alt (?v):
PQ: ?v is neural
?v is not the primary variable
Ensure-student-knows (has-mechanism(?v, neural))
Ensure-student-knows (current-stage(DR))
Ensure-student-knows (has-value(?v, DR, no-change))

[6] A CI-hint is so named because it 'conveys information.'

In this schema, the middle semantic form conveys the idea that the student is currently working on the DR phase instead of giving any information about that phase. More research is required to identify the conditions under which this schema is preferred.

The following example, which can be analyzed as an instance of *correct-neural-alt* where all of the semantic forms except the middle one are realized as *nil*, provides an extremely concise example of a human-generated hint.

(67) S: TPR increases.
T: This is DR.

(K31:159–160)

The following example can be analyzed two ways. It can be considered either as an example of *convey-neural-alt*, where the second sentence includes the content of both the middle and final semantic forms, or as an example of *correct-neural*, where the phrase "in DR" is part of the realization of the final semantic form. Further research is required to know which interpretation is more useful for mechanized text generation.

(68) T: TPR is a neurally controlled variable…Then what value would you assign to TPR in the DR?

(K3:43)

4.6.6 Building complex corrections

In this section we examine what happens when the student gives a wrong answer to one of the intermediate questions in the *correct-neural* schema. The first example starts with an instantiation of *convey-neural-alt* as an interactive explanation:

Elicit-info:	Ask student for the control mechanism for the variable
Convey-info:	Inform student that the current stage is DR
Elicit-info:	Ask student for the correct value of the variable

After the student gives an incorrect value for HR, the tutor drops the *convey-neural-alt*

schema and retries using *convey-neural* instead. Only the middle form of *convey-neural* is instantiated. The first form is omitted because the tutor has just received direct feedback that the student knows that information, and the final form is omitted because turn-taking rules can be used in a human-to-human conversation. Thus the retry could be considered a hint.

(69) T: I need to remind you. Things work according to the way that they are controlled. How is HR controlled?
S: Autonomic nervous system.
T: This is the DR. How will HR change?
S: MAP changing affects baroreceptor reflex changing, affecting HR.
T: In DR no reflex changes have occurred yet.
S: So HR will not change.
T: Correct…

(K34:180–186)

In the next example, the tutor starts with a completely interactive variant of the interactive explanation pattern:

Elicit-info:	Ask student for the control mechanism for the variable
Elicit-info:	Ask student about the activation of the nervous system
Elicit-info:	Ask student for the correct value of the variable

When the student answers the second *elicit-info* with a nonsense answer ("diastolic relaxation"), the tutor retries that goal using *convey-info*. Since the question answered incorrectly is an intermediate question, this type of retry is a feasible alternative. When the failed question is the final question of an attempt, i.e. asking the student for the value of a variable, such a retry would result in giving the student the answer. In that case the tutor might prefer other options, such as retrying the whole *correct-neural* schema or switching to a different schema.

(70) T: … What input to the heart causes CC to change?
S: Sympathetic stimulation. *(i.e. sympathetic nervous system)*

T: Right. Does sympathetic stimulation change during the DR phase?
S: Does DR mean diastolic relaxation?
T: No! The DR occurs during the period of time before any reflex response to the perturbation of the system takes place.
S: Oh, if I understand the question, I don't think that sympathetic stimulation acts during DR.
T: So, predict what change will occur to CC during the DR period.

(K16:39–45)

In the following example, the tutor gives a negative content-oriented reply, then retries the question. The student gives the correct response ("by neural control") the second time, permitting the tutor to continue with the *correct-neural* schema.

(71) T: In what way is CC controlled?
S: It's controlled by the volume of blood in the compartment and affected by inotropic changes.
T: Not quite. Changing the volume changes the performance of the muscle via the length/tension relationship, i.e. Starling's Law. Changing the inotropic state of the myocardium is what we mean when we refer to CC. By what mechanism is CC controlled, then?
S: By neural control?
T: So how will CC be affected in this DR period?

(K31:60–64)

4.6.7 Correcting subsequent neural variables

For the second or third erroneous neural variable, the tutor tends to refer to the logic just explained instead of repeating it. Even version 2 of CIRCSIM-Tutor, which does not take context into account under most circumstances, recognizes repeated neural variables and generates a shorter dialogue for subsequent ones. The following schema, which always generates an interactive explanation, can be used to generate the text used by our human tutors.

Correct-repeated-neural (?v):
Elicit-info (has-mechanism(?x, neural))
Elicit-info (has-value(?x, DR, no-change))

This schema can be instantiated in two ways, depending on whether or not the concept "in DR" is explicitly represented in the text generated by the second semantic form. Below is an example of each.

(72) T: … What other variable is neurally controlled?
S: CC.
T: Super. So what value would you assign it?
(K4:31–33)

(73) T: … What other variable is under neural control?
S: TPR.
T: Right. So how will it be affected in DR?
(K5:35–37)

Human tutors can combine these goals into a single turn, resulting in concise pieces of tutoring such as the following:

(74) T: … What other neurally controlled variable is there and how is it affected?
(K6:25)

(75) T: … While we're on the subject, what other variable is under neural control and how will it be affected in the DR?
(K5:33)

By using suitable prerequisites and realization rules, the schema given in Section 4.6.1 for correcting the first neural variable could be generalized to cover all neural variable corrections. Whether that option is more convenient is a programming issue.

4.7 Correcting non-neural variables

4.7.1 Getting a value via the use of determinants

Version 2 of CIRCSIM-Tutor used the schema below to correct all non-neural variables. Although it is an effective teaching strategy, using it as the sole teaching strategy and realizing it with the same words every time makes text generated by version 2 sound tedious and repetitive, and deprives the student of the deeper learning experience

which would be provided by a richer conversation.

Correct-via-main-determinant (?v):
PQ: ?v is not neural
?v is not the primary variable
Get-main-determinant (?v, ?determinant)
Get-relationship (?v, ?determinant)
Get-value (?v)

The following text shows an example of this pattern as implemented in version 2.

T: What determines SV? *(i.e. main determinant)*
S: CVP.
T: And what is the relation between CVP and SV?
S: Direct. *(i.e. directly proportional)*
T: So what is the correct value of SV?
S: It decreases.
T: Yes, it must decrease.

(CIRCSIM-Tutor v. 2[7])

Although the ideas behind this pattern permeate the transcripts, this pattern itself is relatively rare because other options can be used to generate more concise text. For example, the tutor can switch to a deeper domain model (Section 4.10.1), use the *show-contradiction* pattern (Section 4.5.4) or use the *move-forward* pattern described in the next section. These options generate more cohesive conversation because they permit the tutor to make use of the values of variables obtained in earlier steps instead of requiring the tutor to ask again for the value of a variable which has already been mentioned. The use of *move-forward* is especially pronounced in the early transcripts, where, due to the protocol in use, it was likely that the desired determinant had just been mentioned.

A more general pattern for correcting a non-neural variable is given below. In the pattern above, the tutor asks the student to identify the main determinant so that only one

[7] This text was generated in the RR stage of problem 3 (CC decreases) by giving correct values for all variables except SV.

determinant need be evaluated, while in the following pattern, the tutor requests the value of all the determinants and then asks the student how to combine them. The knowledge used by the TIPS planner includes representations of potential dialogues rather than simply representations of the rules so that alternatives like these can be made available.

Correct-via-determinants (?v):
PQ: ?v is not neural
?v is not the primary variable
Ensure-student-knows (has-set-of-determinants (?v, ?s))
For each determinant ?d in ?s
Teach-info (has-value(?d, DR, ?val))
Derive-value-from-determinants (?v, ?s)

The form *teach-info* is used instead of *ensure-student-knows* to ensure that some text will be generated even if the values of the determinants had been discussed earlier in the conversation. Like the preceding pattern, this pattern is rarely found in the transcripts in its entirety, but the lower-level routine *derive-value-from-determinants* is sometimes invoked by the tutors. The following two examples illustrate three instances of *derive-value-from-determinants*.

(76) T: So since CC (one determinant) is unchanged and the other one increased, what must happen to SV?

(K28:10)

(77) → T: Well, you made predictions about how RAP and CC would change as a result of the pacemaker malfunction. What do you think will happen to SV?
S: It doesn't change.
→ T: Well, you predicted that RAP would in fact go down and you predicted that CC would not change. So what happens to SV?

(K14:51–53)

The following example shows how *correct-via-determinants* can be successful even though a number of nested subschemata were required before the first subgoal could be

satisfied.

(78) T: … First, what are the determinants of RAP?
S: *(after several failed attempts)*
Then it's CO.
T: Super! Now since we know how the CO changed, what will the change in RAP be?

(K30:64–70)

For the variables CO and MAP, instead of asking the student for the determinants, the tutor has the option of asking for an equation giving the value of the variable. The equations referred to are CO = SV * HR and MAP = CO * TPR.

4.7.2 Moving forward along an arrow in the concept map

Move-forward is a version of *correct-via-determinants* which applies when there exists exactly one determinant, that determinant happens to be the last variable discussed, and *teach-info* is instantiated with *convey-info*. The purpose of *move-forward* is to change the topic of conversation from one variable to the next one, following a logical solution sequence for the problem. In the early transcripts, tutor and student converse after every variable, which makes *move-forward* more common than it is likely to be in CIRCSIM-Tutor. The following excerpt shows a typical example of *move-forward*.

(79) T: … Now when CO falls, how does that affect RAP?

(K31:84)

The following example shows two invocations of *move-forward*, one ending with *convey-info* and one with *elicit-info*.

(80) T: … Since CO goes up early in the response, that will cause RAP to fall. Now, what will happen to SV?

(K2:32)

The following example of *move-forward* invokes a deeper level of the concept map.

(81) T: When MAP increases, it's harder for the ventricle to pump blood. So what would that do to SV?

(K31:80)

In the examples above, the tutor identifies both variables on the solution path. In the example below, the tutor identifies the earlier variable, then requests the second variable from the student.

(82) T: … Now let's go back to CO. You said that it would increase. What variable would that affect?

(K4:35)

In each of the following examples, the tutor uses two instances of *move-forward* in succession to move through the concept map.

(83) T: If HR is increased, which variable will be immediately and directly increased by this change?
S: CO.
T: Right. And if CO is up what will change next?

(K48:62–64)

(84) T: … If SV falls, what parameter would that affect?
S: CO falls.
T: Great. Now, when CO falls, how does that affect RAP?

(K31:82–84)

In the following example, the student responds with both variables and values although only variables were requested. By accepting this opportunistic response, the tutor avoids the necessity of requesting the values as a separate step.

(85) T: I'd like you to think about some of the other variables in the table. Especially variables that are immediately and directly determined by HR…
S: HR increases and CO increases.
T: Great…

(K5:21–23)

4.8 Other methods for correcting variables

4.8.1 Primary variable tutoring

The term "primary variable tutoring" refers to tutoring the student about the identification of the procedure variable (first variable affected), the primary variable (first core variable affected), and their direction of change according to the rules in Section 2.4.2. Since these rules are similar to the propagation rules tutored in the previous two sections, the same or similar schemata can be used to tutor them, although primary variable tutoring often invokes a deeper level of the concept map. In the following example, the tutor requests the primary variable and its direction separately. The tutor's response includes a positive content-oriented reply ("The vascular smooth muscle…") which supports the student's answer.

(86) T: OK, then predict the first variable that would be affected by injecting the alpha agonist.

…

S: TPR.

T: Sure. The vascular smooth muscle contains alpha receptors. In what direction will TPR change?

(K32:18–22)

In the following example, the tutor requests the primary variable and its direction at the same time. After the student gives the wrong direction, the tutor uses a negative content-oriented reply to hint at the correct direction. In the terminology of Section 4.5.2, the tutor is supporting the negation of the student's statement.

(87) T: Begin by predicting the first variable that is affected by the pacemaker malfunction. And the direction in which that variable will change.

S: HR decreases.

T: No. The pacemaker went from 70 to 120 beats per minute.

S: HR increases.

(K7:15–18)

In the following excerpt, the tutor uses a negative content-oriented reply to redirect the student's attention. In this example the tutor is denying prerequisites for the student's statement to be true.

(88) T: OK, then predict the first variable that would be affected by injecting the alpha agonist.
S: CC.
T: The myocardium only contains beta receptors and hence can't respond to an alpha agonist.
(K32:18–20)

The following example shows a typical construction used by the human tutors. We do not expect to use yes/no questions in CIRCSIM-Tutor because they do not give as much information as other types of questions.

(89) S: RAP decreases.
T: Is RAP the first variable that will be affected by the pacemaker malfunction?
S: No.
T: Then what is the first variable that is changed?
S: HR.
T: And how will it change?
S: HR increases.
(K15:16–22)

4.8.2 Pointing to the answer

According to Hume et al. [1995], the tutor can "point to" the answer as an alternative to conveying information. Hume et al. call this pattern a PT-hint (PT = 'point to'). The following example shows the frequent usage of this pattern in version 2 of CIRCSIM-Tutor.

T: ... What is the correct value of SV?
S: Up.
T: No, SV does not increase.
→ Consider the value of RAP.
What is the correct value of SV?

(CIRCSIM-Tutor v. 2)

This pattern is not common in the transcripts. The following excerpt, where the tutor mentions a noun without giving information about it, is an example of the pattern.

(90) T: … Think about the effect of the change in MAP.

(K54:2)

The following examples, where the tutor responds with a question instead of a declarative statement, are also potential examples of this construct.

(91) T: … How about the influence of a change in CO on RAP?

(K11:65, cf. Hume et al. [1995], example 1)

(92) S: Then MAP would increase even more.
→ T: What does the baroreceptor reflex do?

(K31:101–102, cf. Hume et al. [1995], example 9)

4.8.3 Giving the student the answer

Although the human tutors do not like to do it, it is possible to give the student the answer if further dialogue seems like a waste of time. Thus *give-answer* is a simple schema which contains the semantic form *convey-info*. The following example shows *give-answer* used during primary variable tutoring.

(93) T: In this example, the first variable (in the predictions table) to change is RAP. Its effective change is to decrease (relative to what happens in the ventricle)…

(K20:98)

4.9 Conveying and eliciting information

This section discusses a number of ways to implement the *convey-info* and *elicit-*

info forms which are invoked by higher-level schemata and used in addition in content-oriented replies.

4.9.1 Conveying a definition

There are several kinds of domain knowledge, including definitions, facts and rules, which can be conveyed to the student by roughly similar methods. Since the knowledge incorporated in a definition is usually part of a lower-level concept map, it is probably better to restrict CIRCSIM-Tutor's ability to use this tactic until it can understand any terms it introduces. In this section we show several examples of conveying a definition to the student.

Anatomical and physiological concepts can be defined in a declarative sentence, sometimes nested inside a *remember* clause or a similar construct:

(94) T: Remember CO is a measure of the rate at which blood is being taken from the central venous compartment.
(K33:164)

(95) T: [Alpha receptors] cause an increase in intracellular calcium concentration and thereby trigger the effects of the sympathetic nerves or whatever cells have the receptors in their membranes.
(K31:20)

The definition of the DR stage occurs frequently because it is a common way to instantiate the *current-stage* subgoal of the *correct-neural* schema.

(96) T: … DR is defined as the period of time before any reflex activity can occur and hence before the sympathetic nervous system could change its firing…
(K26:58)

(97) T: … DR is before there are any neural changes…
(K5:39)

(98) T: Reminder, the DR occurs before there are any reflex (neural) changes…
(K3:41)

(99) T: … The DR occurs during the period of time before any reflex response to the perturbation of the system takes place.

(K16:43)

In the following example, the definition is nested inside two enclosing clauses, one for *remind* and one for *predict*. However, from the point of view of CIRCSIM-Tutor, the *predict* clause does not contribute to the semantics of the sentence. Thus we may be able to generate it as an alternative to the options above during lexical insertion without needing to understand the details of this use of *predict*. (Of course, we still need *predict* in the lexicon for other purposes, such as asking the student to predict a value.)

(100) T: … Now I would remind you that you are predicting what would happen before there are any reflex changes. Would you like to change any of your predictions?

(K31:56)

Sometimes, as in the following example, the form of the definition preferred by our experts violates the semantics of a word as it is usually used in CIRCSIM-Tutor. For example, although cardiac output is usually a measure, it is equated to a process in the following excerpt.

(101) T: … CO is a process in which venous blood is transferred to the arterial tree…

(K2:28)

Two cases where this frequently occurs are the definitions of *preload* and *afterload*.

In the following example, the tutor has incorporated a definition and a request for a value into one sentence.

(102) T: What I was asking is what determines how much blood is ejected from the heart each time it beats (i.e. the SV).

(K14:49)

4.9.2 Conveying a fact

Facts are a second kind of domain knowledge which the tutor may wish to convey to a student. The following examples show some ways the tutor can convey a fact from the domain knowledge base.

(103) T: No. [TPR] is neurally controlled. Try again.
(K7:27)

(104) T: Remember, [TPR] is a neurally controlled variable. The reflex is responding to an increase in MAP in the DR. Try again.
(K7:73)

(105) T: No, [CC] is under neural (sympathetic) control…
(K5:31)

In the following excerpt, the tutor refers to a task-specific fact. The mechanism used is identical.

(106) T: … We're talking about what happens before there are any neural changes.
(K11:51)

4.9.3 Conveying a rule

Rules are a third kind of domain knowledge which the tutor may need to convey. By stating a rule, the tutor is trying to convey one of the underlying goals of the system directly instead teaching it by example. In the following example, the tutor is trying to convey rule *algebraic-SS* from Section 2.4.4.

(107) T: … When you have a situation where a parameter changes in one direction in DR and in a different direction in RR, usually (not always) the DR change is greater…
(K33:126)

Although the current protocol does not permit the tutor to control the order in which variables are predicted (see Section 3.3.2), the tutor can teach the student about the

importance of a logical prediction order.

(108) T: … Before we go on, I want to call to your attention something you did in making your predictions. You predicted a value (change) for RAP before you knew what the determining variables had done. That's not possible. You need to proceed causally in making your predictions…
(K32:64)

The methods in this section can be listed in sequence from most common to least common as follows:

- Give a definition
- Convey a domain fact
- Convey a domain rule
- Convey a task-specific fact

For text generation, however, what is important is not the relative frequency but the conditions under which each type of semantic forms is chosen.

4.10 Switching between domain models

4.10.1 Temporarily switching to a deeper domain model

If the student's response is a step on the correct path, one option is to temporarily switch to a view of the cardiovascular system which includes the concept just mentioned by the student. This concept may be a variable from a deeper-level concept map or an item from a functional model. Once the student understands the concept in terms of the deeper domain model, the tutor attempts to return the conversation to the core variables, either as part of the new schema or by returning to the original schema.

In the following example, the tutor wants the answer "nervous system" as the determinant of TPR. The answer "radius of arterioles" is a step on the correct path. Since it is not on the top-level concept map, the tutor switches to a lower-level concept map,

replacing the goal *elicit-info (has-mechanism(TPR, neural))*, or its equivalent:

Elicit-info (determines(nervous-system, TPR))

by the two-part schema:

Elicit-info (determines(radius-of-arterioles, TPR))
Elicit-info (determines(nervous-system, radius-of-arterioles)).

The first part of this schema has already been satisfied by the student's response. By responding correctly to the second part, the student will have answered the original question and returned to the top-level concept map, allowing the tutor to continue with *correct-neural*.

(109) T: What is the primary mechanism of control of TPR?
S: Radius of arterioles.
→ T: Yes. And what is the primary mechanism by which arteriolar radius is controlled?
S: Sympathetics. *(i.e. sympathetic nervous system)*
T: Yes. And we're dealing with the period before any change in nervous activity occurs. So what do you think about TPR now?
S: It stays the same.
T: Correct…

(K12:37–43)

The following example is similar.

(110) T: Yes and what variable would [MAP] affect directly?
S: Afterload.
T: Super. And the afterload does what to which variable?
S: Decreases SV.
T: Great. Correct. So SV goes down and as you said, this causes CO to fall. What effect would this have on RAP?
S: It increases.

(K32:50–63)

In the next example, the student gives a more detailed answer which is incorrect. The tutor gives the student the correct answer at the same level of detail, stating that size of the

arterioles is more important than pressure, then continues as in the previous example, asking for the determinant of size. The correct answer to this question brings the student back to the top level of the concept map. Now the tutor can continue with the *correct-neural* schema.

(111) T: How is TPR controlled?
S: Besides pressure at the pre-capillary, or arteriole level?
T: TPR is determined primarily by the size of the arterioles. In blood pressure regulation (our present topic of discussion) how are changes in arteriolar size brought about?
S: O.K. Autonomic nervous system innervation.
T: Correct. So what direction would TPR change in the DR period?
(K34:68–72)

In the following example, the tutor again wants the answer "nervous system" as the determinant of CC. The tutor asks a diagnostic question, then chooses a part of the student's answer which answers the original question. This allows the tutor to return to the top-level concept map and use *derive-value-from-determinants* to elicit the value of CC. In CIRCSIM-Tutor, we only generate such questions when we expect to be able to understand the answer.

(112) T: Do you know what physiological inputs determine [the value of CC]?
S: Calcium.
T: How does calcium determine contractility?
S: A direct ratio of the amount of calcium to excite the cardiac muscle fibers. Along with the sodium channels and [the sympathetic nervous system].
T: You're right, changing sympathetic inputs to the heart does change contractility by varying … Now, if CC is determined by the sympathetic nervous system, what change in CC will occur in DR?
(K44:62–66)

Usually a shift to a deeper model is precipitated by the student, but occasionally it is the tutor who shifts to a new model. In the following example, the tutor introduces deeper information and uses it as the basis of a question, then uses the correct answer to the

question to return at the top level.

(113) S: RAP increases.
T: RAP is correct. But "increase" is a problem. Let me remind you about the vascular function curve…
…
T: The vascular function curve relates CO and central venous pressure.[8]
…
T: So when CO increases, what happens to central venous pressure?
S: It decreases.
T: Great. What variable has the same value as central venous pressure?
(K6:28–35)

In the following example, the tutor decides to move to a functional model after failing to elicit the determinants of RAP. We label the marked statement as referring to a functional model instead of a deeper-level concept map because it describes an abstract concept, *any compliant structure*, rather than a specific anatomical structure. The tutor's comments here are part of a larger structure for teaching the relationship of RAP and CO; the rest of the argument has been instantiated as *nil*.

(114) S: *(gives wrong determinants for RAP)*
T: Try to look at it this way. The right atrium is like the end of the venous system. And the veins are very compliant.
…
→ T: The pressure in any compliant structure is determined by the volume contained by that structure and the compliance of that structure.
S: Then it's CO.
T: Super!…
(K30:65–70)

[8] This transcript is from a protocol where RAP is a core variable and central venous pressure (CVP) is not.

4.10.2 Moving to a higher-level model

In the previous section, the new schema ended by inquiring about the same variable as the original schema. If it does not, the tutor must explicitly try to move the conversation back to the core variable being corrected.

In the following example, the tutor wants to correct the value of SV. Since one of the determinants supplied by the student is not stated in terms of a core variable, the tutor must elicit the name of the corresponding core variable before continuing with the outer schema.

(115) *(the beginning of this transcript is missing, but it is likely that the student said that the determinants of SV are CC and preload right before the system crashed)*
T: To continue, determinants of SV are CC and something related to preload. Do you know what is meant by preload?
S: Preload refers to the amount of blood available in the left ventricle before systole.
→ T: Sure. What variable in our list reflects this?
S: RAP.
T: Right. And what happened to RAP?
S: It increased.
T: So since CC (one determinant) did not change and the other one increased, what must happen to SV?
S: It must increase.
(K28:3–11)

The following example is similar. The tutor accepts the student's response but immediately directs the conversation back to the corresponding core variable.

(116) S: Determinants for SV are CC and filling of the left ventricle.
T: Right. What parameter here reflects filling of the left ventricle?
(K27:65–66)

In the following example, the tutor restates the student's statement with the aim of moving the conversation back to the top level. In each of these examples, from a linguistic point of

view, the tutor's utterance might be considered a response, but from a pedagogical point of view we consider it to be new material since it serves the goal of returning the conversation to the top-level concept map.

(117) T: … How is TPR controlled?
S: Sympathetic vasoconstriction.
→ T: Right. TPR is primarily under neural control…

(K11:49–51)

4.11 Generation of previously observed phenomena

Previous students of the CIRCSIM-Tutor transcripts have analyzed them with respect to a variety of phenomena. These phenomena include surface grammatical and lexical phenomena as well as psychological criteria such as the analyst's belief about what the student is thinking. For text generation purposes, we must define semantic forms which represent the tutor's intentions. Thus surface phenomena cannot be used as a basis for analysis, as a particular surface phenomenon may be generated by more than one semantic form.

An additional disadvantage of using surface phenomena to classify tutorial patterns is the fact that text can undergo a variety of transformations before it is uttered. In the mechanized tutor, these transformations are part of the turn planning step. Since multiple semantic forms may have been combined to generate one sentence, an underlying representation must be posited if we are to have an accurate representation. For example, a hint generated by *convey-info* and a request for a new value of a variable generated by *elicit-info* may be combined into one surface question. Unless we have posited an underlying representation, results obtained by counting instances of surface structures cannot give accurate data about the planning process. In this section we examine tutorial phenomena defined by others and demonstrate how we would generate the same text.

4.11.1 Summaries

The term *summary* has been used to describe a number of distinct phenomena. In the most common case [Sanders 1995], the term is used to refer to a portion of the solution to a procedure, as in the following example:

(118) T: … So let's see where we are. HR went up and therefore CO went up. CO went up and therefore MAP went up…
(K3:31)

To generate a text like this, we use a series of instances of *convey-info*, one for each step in the problem-solving procedure. To obtain a summary, each instance must be instantiated as a declarative sentence. If some of them are instantiated as questions, we obtain an interactive explanation, as shown in the example below. Thus this type of summary is a special case of a more general review mechanism.

(119) T: And if HR goes down, what happens to CO?
S: CO decreases.
T: And if CO is down, what happens to RAP?
S: RAP will increase.
T: And if RAP is increased, what will happen to SV?
(K36:96–100)

Hume et al. [1995] also use the term *summary* to describe the statement of a rule, as in the following example.

(120) T: … Let me summarize. There are three neurally controlled CV effectors: CC, HR and TPR. If the stimulus that we apply to the system doesn't act directly on any of them none of them will change in the DR…
(K34:74, cf. Hume et al. [1995], dialogue 1)

This excerpt occurs at the end of DR for a student who had gotten all of the neural variables wrong. The tutor had already corrected each of the neural variables individually.

In the following example, the term *summary* is used to describe a regular

instantiation of one of the subgoals of *correct-neural* preceded by the cue phrase "we just said that …".

(121) T: … But we just said that were dealing with a period in which there are not yet any changes in neural activity…
(K34:58, cf. Hume et al. [1995], example 6)

Finally, the term *summary* can refer to a restatement of previous items mentioned in the conversation, as in the example below. This mechanism is one we have not studied yet.

(122) T: … Here's what we know and agree on: CC doesn't change; MAP increases…
(K31:76, cf. Hume et al. [1995], example 5)

4.11.2 Explanations

The examples below show three excerpts labeled *explanations* by Hume et al. [1995]. We would generate the following example with *give-answer*.

(123) T: … What happens to central venous pressure when CO increases?
S: *(gives a long answer)*
→ T: Well you're partly correct. When CO increases, it does so at the expense of the central blood volume (venous volume). During that period, CO exceeds venous return. When the venous return finally catches up, CO equals venous return. However venous return never exceeds CO so the veins cannot be refilled without reversing the original process, i.e. reducing CO.
(K9:27–29, cf. Hume et al. [1995], dialogue 2)

The next two examples are content-oriented replies, one positive and one negative. They might also be tabulated as hints.

(124) T: … Now lets talk about your predictions. First, what are the determinants of RAP?
S: TPR.
→ T: Not really. Try to look at it this way. The right atrium is like the end of the venous system. And the veins are very compliant.
S: Is it the MAP then?

→ T: No. The pressure in any compliant structure is determined by the volume contained by that structure and the compliance of that structure.
(K30:64–68, cf. Hume et al. [1995], example 4)

4.11.3 "Directed lines of reasoning" (DLRs)

Sanders [1995] defines a *directed line of reasoning* as a standard series of questions which the tutor uses to teach a specific point. If the student does not give the expected response at any point in the series, the tutor must abandon the line of reasoning and try a different tactic.

From a planning perspective, a DLR or interactive explanation is generated when some or all of the subgoals making up a schema are realized as questions. With respect to text generation, there is no difference between the recurrent two- and three-turn sequences which are among the human tutors' most frequently used mechanisms and the longer and more complex ones which are usually labeled DLRs.

The following example can be generated from a series of instances of *move-forward* where each instance has been realized as a question.

(125) T: … Now considering that the first things that are going to change are the things that are under neural control, which of these determinants would be the first affected?
S: CC.
T: Of course! And in what direction?
S: Decrease.
T: Right again. And how would that affect SV?
S: Decrease.
T: Sure. And what effect would that have?
S: Decrease CO.
T: Yes again. Then what?
S: MAP decreases.
T: Yes, again. And in this regard it is MAP that is regulated by the BAROreceptor reflex. That's why it's called that…
(K12:65–74, cf. Hume et al. [1995, example 3], also cited by Sanders [1995, p. 94])

4.11.4 Hints

Hume et al. [1995] define a *hint* as any response by the tutor after the student gives an incorrect answer to a question, unless the tutor gives the correct answer. This definition, which includes phenomena which can be cross-classified as summaries, explanations, interactive explanations and negative acknowledgments, is much broader than the everyday use of the word "hint." The phenomena classified as hints by this definition can be generated by almost any combination of semantic forms. For example, Hume labels the marked turn below as a CI-hint (CI = 'convey information'):

(126) T: How would HR and CC be DIRECTLY AFFECTED by the administration of the alpha agonist?
S: Would the heart rate decrease because it is not getting enough oxygen because the coronaries are constricted?
→ T: No, the local metabolic control of the coronaries is more powerful than the neural vasoconstriction that the alpha agonist is producing. HR and CC are determined by the nervous system. In DR all neural effects haven't happened yet, so predict what will happen to them.
(K35:102–104, cf. Hume et al. [1995], dialogue 3)

We would generate this turn using the following series of semantic forms:

- Negative acknowledgment: "No."
- Rebuttal: "the local metabolic controls … producing."
- An instance of *correct-neural*:
  - Subgoal *has-mechanism* realized using *convey-info*.
  - Subgoal *current-stage* realized using *convey-info*.
  - Request for value realized using *elicit-info*.

Dictionary definitions of 'hint' depend on terms such as "indirect suggestion" and "slight indication." Since such definitions depend on human judgment, it is difficult to use them to determine which textual phenomena should be considered hints. For example, should sentences generated by the first two subgoals of the *correct-neural* schema ("HR and CC are determined by the nervous system. In DR all neural effects haven't happened

yet") be considered a hint? They sound like a hint in this context, but if they appeared alone we might consider them an explanation.

We showed above that many CI-hints can be generated by choosing a variety of realizations for the semantic form *convey-info*. CI-hints and PT-hints are generated by separate schemata; the latter are considered briefly in Section 4.8.2. Sections 4.5.2 and 4.11.2 include examples of hints which appear to belong to the content-oriented reply section of the turn.

4.11.5 Negative acknowledgments

Spitkovsky [Spitkovsky 1992; Evens et al. 1993] identifies ten categories of negative acknowledgments. According to his categorization, the term "negative acknowledgment" includes any utterance where a hearer could infer the tutor's belief that something the student said was incorrect. This definition is too broad to be useful for text generation. We only want to label text as a negative acknowledgment if that was the tutor's direct intention.

Two of Spitkovsky's cases are generated as negative acknowledgments in our classification. In the following examples, one negative acknowledgment has been realized as a particle and the other as a complete sentence.

(127) T: Do you know a formula that gives you a deterministic statement about MAP?
S: Diastolic volume minus end systolic volume.
→ T: No, that would tell you stroke volume.
(Spitkovsky [1992], category 1)

(128) S: … because your body always needs the same amount of blood with oxygen.
→ T: Well, first of all, what you said isn't really correct.
(Spitkovsky [1992], category 2)

In the following two examples, the tutor uses a negative content-oriented reply to correct something the student has said.

(129) T: *(asks which parameter student will predict next)*
S: SV.
→ T: In order to predict a parameter, you have to have predicted its determinants…
(K25:20–22, cf. Spitkovsky [1992], category 3)

(130) T: What does CO do to the volume of blood in the central venous compartment?
S: It's increasing it.
→ T: It's increasing it? It seems to me that every time the heart beats, it's pulling a stroke volume of blood out of the central venous compartment.
(Spitkovsky [1992], category 5)

In the following two examples, the tutor wants to question something the student has said. We would generate the first as a pseudo-diagnostic question and the second using the *show-contradiction* schema.

(131) T: Well, what is the reflex attempting to do?
S: It's attempting to lower HR, I would imagine back to normal.
→ T: Is it HR that's under control?
(Spitkovsky [1992], category 6)

(132) T: You predicted that CO would stay the same. You predicted TPR would go up. You predicted MAP would stay the same. Is that possible?
(Spitkovsky [1992], category 7)

The following excerpt shows a diagnostic question. We generally avoid generating open-ended questions such as this one because of the burden they place on the input understander.

(133) T: What did you predict CO would do?
S: I would say it wouldn't change.
→ T: Why do you say that?
(Spitkovsky [1992], category 8)

In the following case, the tutor gives the student the answer rather than continuing to explore the question. Although the *give-answer* schema most often occurs in conjunction with an explicit negative acknowledgment, it need not.

(134) T: Do reflexes fully compensate for a disturbance?
S: I don't know. That's what I was predicting, but maybe not.
→ T: They do not.

(Spitkovsky [1992], category 10)

The final example shows the motivation for generating text a turn at a time in the mechanized system instead of one semantic form at a time. In this example, the pressure has two determinants. The tutor's response could be generated by a mixed acknowledgment followed by an elicitation of the missing determinant. In this example, the tutor combined these semantic forms to generate the concise response shown. Equivalent but less concise responses would include combining the two forms as in "What is the other determinant?" or uttering separate sentences, as in "There is a second determinant. What is it?".

(135) T: What determines the pressure in the central venous compartment?
S: The amount of blood that's in there.
→ T: And?

(Spitkovsky [1992], category 9)

4.12 Problems with cooperative conversation

A *cooperative conversation* can be defined as a conversation where each speaker has goals to achieve and a plan to achieve them. This definition distinguishes a cooperative conversation from one where one agent has a goal and a plan to achieve it, and the other's job is mainly to go along with the plan.[9] A fully cooperative conversation can be compared

[9] The subordinate agent is still being cooperative in the sense of Grice [1975].

to two children building a tower of blocks together, as opposed to one child telling the other what to do. In the cooperative case, neither may be able to predict the shape of the resulting tower.

Most conversations between humans and computers are led by one party or the other. For example, in a database front-end or an ITS used for exploring a topic, the human user is likely to be asking questions, and the program is just answering. Conversely, in an advice-giving system or a program such as an automatic teller machine, the program leads and the human's job is to respond. Although some programs, such as a database system which uses forward reasoning, may appear to be contributing actively to the conversation, such a conversation is not in fact a plan-based activity.

Many human-to-human tutoring sessions are also mainly led by one party or the other. In this regard, one might contrast our experimental setup, where the tutor has an agenda to fulfill, with the tutoring sessions described by Fox [1993], where the student chooses the agenda for the tutoring session.

Although many researchers have studied multiple-agent conversations, it is not yet feasible for a broad-coverage ITS to deal with the situation where both entities have plans. In addition there is the increased difficulty of understanding exchanges begun by the student. In the absence of a cooperative conversation model, the need for plan recognition or general student initiative processing does not arise.

The following conversation fragment shows some examples of cooperative phenomena which are outside the range of our model.

(136) T: What other variable is under neural control primarily?
S: CC?
T: Yes. You predicted that it would go up. Still feel that way?
S: Yes.
→ T: But remember that we're dealing with the period before there can be any

neural changes. How can CC go up if it's under neural control?
S: *(explains how he/she believes CC can go up using terms from deeper concept map)*
T: *(attempts to deal with student's confusion)*
Do you see the difference?
S: No, this concept is hard for me to grasp.
T: *(explains further)* OK?
→ S: Is increased calcium the only thing that can increase contractility?
T: Yes.
S: OK. So would it be accurate to say that *(asks a follow-up question)*?
T: Yes and *(answers follow-up question)*. OK?
S: OK.
T: So what's your prediction of CC in the DR?

(K10:39–53)

In the first marked statement, the tutor encourages the student to give an explanation in terms of a deeper-level concept map via an open-ended question. Starting with the second marked statement, the student takes the initiative, referring to a concept mentioned in the answer to the earlier question and which is now in the shared mental model of tutor and student.

The following excerpt shows some additional examples.

(137) T: … Can you tell me how TPR is controlled?
S: Autonomic nervous system.
T: Yes. And the predictions that you are making are for the period before any neural changes take place. So what about TPR?
→ S: I thought TPR would increase due to higher flow rate through vasculature.
T: You just said that the primary control over the TPR was via the autonomic nervous system. The autonomic nervous system activity would not have had time to change yet in the DR period.
→ S: OK. I was thinking about TPR intrinsically and extrinsically. So the autonomic nervous system would affect the system extrinsically and control it but wouldn't there be more friction on the fluid going through the tube?
T: TPR is a function of the extent of contraction of the vascular smooth muscle. That determines the vascular radius, present in the resistance equation for each blood vessel as an inverse fourth power function. Sure, increasing the flow by increasing the pressure gradient would occur but the

calculated TPR wouldn't change.
S: OK.

(K10:29–36)

CIRCSIM-Tutor can handle a student response like the first marked example. Although we cannot make use of the explanation given by the student, the human tutor does not make use of this information either. The second marked example is different. Understanding the student's response is beyond the state of the art in natural language understanding. Although a program could be taught to interpret this statement, it is not currently possible for a system to understand this level of complexity on a general basis. There are two principal reasons for this difficulty. One is the abstract nature of the phrase *intrinsically and extrinsically*, a meta-statement which refers to explanations which will occur later in the discourse. The other problem is lining up referents for the abstract physics in the last sentence. The human tutor knows that the student is considering blood in the peripheral arterial system as an instance of a fluid in a tube and uses this information to form a response. Since the mechanized tutor cannot understand this statement, we are obliged to come up with other ways to generate a useful response.

This example is relatively easy for expert human tutors since they can understand the student's erroneous ideas. CIRCSIM-Tutor will have even more difficulty when the student's language is logically confused or does not map well into the mental model used by the program. Borchardt [1994] demonstrates the difficulties of understanding causal reasoning even when the use of natural language is not an issue. Even in the tutor-led conversations which we expect to conduct with CIRCSIM-Tutor, we can reduce the burden on the input processor by using short-answer questions instead of open-ended ones and by ending turns with explicit questions instead of relying on the student's ability to follow turn-taking rules. This topic is discussed further in Section 5.3.7.

Chapter 5
Architecture of the Discourse Planner

What is the real function, the essential function, the supreme function of language? Isn't it merely to convey ideas and emotions?
—Mark Twain

This chapter describes an architecture for the TIPS text generator which will enable CIRCSIM-Tutor to generate conversations based on the model developed in the last chapter. In Section 3.2 we looked at the shortcomings of CIRCSIM-Tutor v. 2 and what would be required to ameliorate them. In Chapter 4 we looked at empirical data from transcripts of expert tutors. In Chapter 1 we looked at earlier work in text generation and the design of ITSs. In this chapter we pull this information together to design a system which will be a qualitative advance over previous work. A detailed set of goals for v. 3 is CIRCSIM-Tutor is listed in the Introduction.

The discourse planner contains two main components, the tutorial planner and the turn planner. We begin this chapter by describing the two principal knowledge representations used in the planner, the plan operators which represent the schemata isolated in Chapter 4, and the semantic forms which are the intermediate representation between the tutorial planner and the turn planner. We describe the architecture of the planner, including the control flow of the system and the function of each component. We give examples of knowledge representation and control flow for both normal and exceptional cases.

5.1 Representation of tutorial knowledge

5.1.1 Conceptual overview of the planning process

In the previous chapter, we abstracted textual schemata from the discourse of expert tutors. In this chapter, we demonstrate how to generate dialogues which are structurally similar to those in Chapter 4. Each schema is represented as a plan operator. In this section we show how these plan operators can be used to move from an analysis of the tutoring situation to a decision about text to be uttered.

The following decisions must be made for each turn to be generated by the tutor.

- Choice of schema
- Choice of level of interactivity
- Choice of pragmatic, syntactic and lexical elements

In choosing the correction schema, the tutor can use any available information about the tutorial situation. A list of potentially relevant factors is included in Section 5.3.4. These factors must be stored in the student model for future reference. Note that a schema might do something simpler or more complicated than teaching one concept. For example, we might know the correct thing to say to the student at a particular point in the conversation without being able to give the concept a meaningful name.

There is an intimate relationship between the knowledge we use to make instructional planning decisions and the student model. If our discourse plans were directly related to the concepts we wanted the student to learn, such as the fact that neural variables don't change in the DR stage, we would need to keep track of whether or not we had taught this concept or believed that the student has learned it. But our schemata describe ways of teaching content rather than the content itself. As a result, it is more important to keep track of which discourse mechanisms we have used and how often. This

method has the additional advantage of giving us information which is useful for increasing the variety of our language. It also permits us to rely more on real knowledge about the student, e.g. the historical record of the student's productions, rather than on postulated concepts such as student difficulties, which can only be inferred from the student's productions.

Once a correction schema is chosen, planning continues until enough primitive semantic forms have been generated to complete the first turn. At some point during this process it is usually necessary to choose the degree of interactivity desired. An abstract semantic form such as *S-knows* or *T-teaches* can be realized using a predicate such as *T-conveys*, which generates non-interactive text, or with a predicate such as *T-elicits*, if more interactive conversation is desired. For example, Section 4.6.1 describes the standard schema for correcting neural variables. Sections 4.6.2, 4.6.3 and 4.6.4 describe the instantiation of that schema to form an interactive explanation, a monologic explanation and a hint, respectively. Although our domain experts prefer active plans involving a lot of questions, if the student is not responsive they may switch to a plan involving more explanations. Note that if a correction schema uses more specific semantic forms instead of the abstract ones used above, it may not be possible to develop multiple variations at this stage. This may be appropriate when attempting to model specific discourse phenomena used by our expert tutors.

Finally, we must determine the surface realization of the turn. The decisions to be made here are somewhat but not completely related. Although we describe this process as a series of sequential choices for ease of explication, it could be implemented in a variety of ways, varying from a close analogue of the description given here to a constraint satisfaction system. First we must decide how to combine the chosen semantic forms into

sentences. For each sentence we must decide what syntactic form to use: declarative, direct question, indirect question or imperative. In Section 5.2.3 we show that this decision is independent of the decisions made in the previous step about the level of interactivity. The following decisions lead step by step toward the creation of a surface form for the sentence: choosing a primary verb and argument structure, choosing lexical items, and creating a grammatical sentence.

Of course, students do not necessary respond the way we expect them to. Sections 5.3.5 and 5.3.6 describe methods for changing the agenda when an unexpected student response occurs.

5.1.2 Motivation for the use of schemata

A major decision which must be made in any text generation system is the tradeoff between the derivation of forms from intentions and the use of precompiled forms such as schemata. Schemata are a logical representation for CIRCSIM-Tutor because they provide a natural model for the patterns identified in Chapter 4. Schemata provide a straightforward and legible way to encode the tutor's goals. They provide a method of connecting our knowledge about pedagogy, i.e. "what to say when," with our knowledge of discourse, i.e. "how to say it." Since having a coherent argument is one of the factors which contributes to coherence in text, the use of schemata also helps to insure coherence in the generated text.

In addition, the schema-based approach offers several more specific advantages for CIRCSIM-Tutor. First, it allows us to control what multi-turn tutoring patterns will look like in the generated conversation. Without the use of schemata, we could not easily guarantee that we could generate the patterns identified in Chapter 4. Second, non-schema based discourse generation rules to ensure coherence in dialogues and other multi-

paragraph texts have not yet been developed. Third, the pedagogical knowledge needed to derive the desired tutorial patterns from domain knowledge alone is not available. As all experienced teachers know, there is a gap between knowing how to solve a problem in a domain and knowing how to teach another person to do so. Finally, schemata require significantly less processing time than other approaches. The use of a schema-based approach is a basic difference between our work and the work of such authors as Chu-Carroll and Carberry [1995], Smith [1992], Horacek [1992], and Moore [1995].

Figures 5.1 and 5.2 show logical derivations of the value of a neural variable and a non-neural variable respectively, using the rules proposed in Chapter 2. These diagrams demonstrate several reasons why some form of discourse knowledge must be represented, i.e. why domain knowledge does not suffice for knowing how to teach something. First, many of the proof steps do not involve cardiovascular physiology at all, but deal with mathematical calculations such as the definition of shortest path. Second, not all of the proof steps shown are equally important for pedagogical purposes. For example, the concept we want to impress on the student is the fact that neural variables don't change in DR. If we tried to derive our language directly from these diagrams, we would clutter up this basic statement with extraneous clauses like "... unless it's primary." Last but not least, there are many ways to teach the student about the concepts represented in these diagrams besides following the proof step by step.

The use of schemata in CIRCSIM-Tutor avoids many of the disadvantages which have been associated with schemata in the past. First, since the primitives of our system are semantic rather than syntactic forms, the use of schemata does not unduly constrain the eventual surface structure. Second, since our planner can switch schemata whenever the student responds in an unexpected manner, the use of schemata does not constrain our

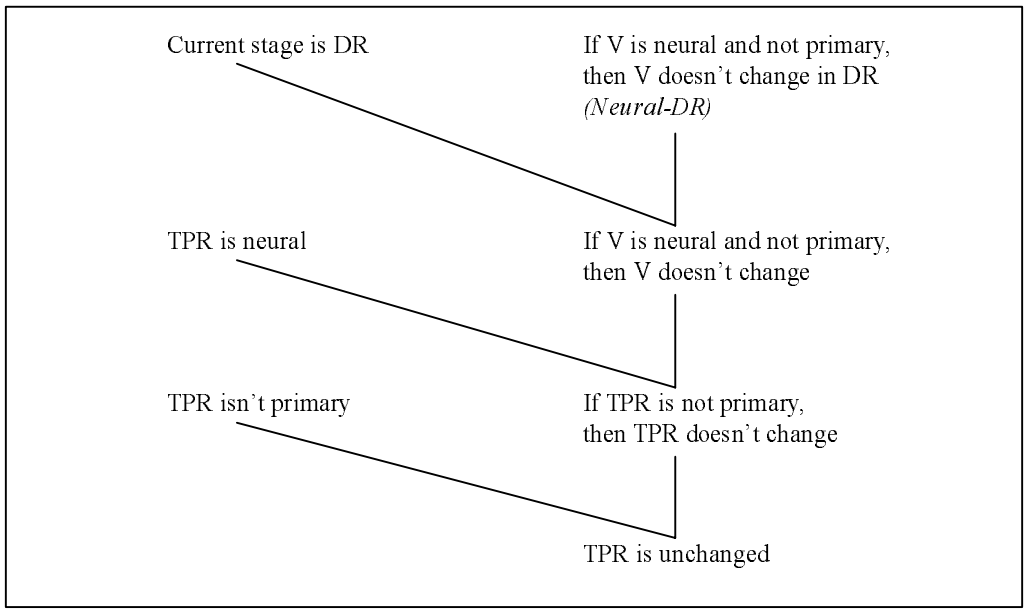

Figure 5.1: Derivation of the value of a neural variable

pedagogical flexibility. Third, our schema syntax is flexible enough to encode the tutorial patterns without difficulty. Finally, since our schemata are encoded as plan operators, each schema can only be used in situations where its prerequisites are satisfied. Thus we can restrict the application of schemata to appropriate contexts.

5.1.3 Categories of schemata

The schemata needed to implement the model proposed in the previous chapter can be loosely grouped into four categories. First, schemata are used to control the hierarchical behavior of the planner at all levels, including the selection and sequencing of variables, the choice of tutorial methods for each variable, and repetitive low-level phenomena. Second, a small number of schemata are needed to model the methods used by our expert human tutors for correcting variables. Since many observed tutoring phenomena differ only in their degree of interactivity, it is possible to use a single dialogue

schema to generate any of several tutoring phenomena, such as a hint, an explanation, or an interactive explanation. On the other hand, since each schema represents one way of teaching a concept, two ways of teaching the same concept, such as a PT-hint and a CI-hint, require different schemata. Third, adjunct schemata, such a schema to correct the student's language or to differentiate cardiac contractility from the Frank-Starling effect, are used to recover when a student gives an unexpected response. Finally, low-level schemata are used to provide multiple ways to instantiate semantic forms. Some examples of the latter are given below in Section 5.2.2.

5.2 The initial set of semantic forms and their realization

5.2.1 Major domain-independent semantic primitives

For CIRCSIM-Tutor v. 3 the choice of which semantic forms to leave as primitive is more a practical than a theoretical one. If the tutorial planner does not need to make a distinction between two concepts, then we can use the same semantic primitive to represent both. Although some work has been done on abstracting semantic primitives from the transcripts, this work is far from complete. In this section, we define some major domain-independent semantic forms which will be needed in CIRCSIM-Tutor regardless of which set of semantic primitives is eventually chosen.

In the following definitions P(x) represents a proposition about an item in the domain knowledge base.

1) *S-knows(P(x))*
   This form is used when we want to make sure that P(x) has been included in the discourse at some point. In a philosophical sense, we cannot know what the student knows, so we use the operational definition of asking whether one of the speakers has stated the concept.

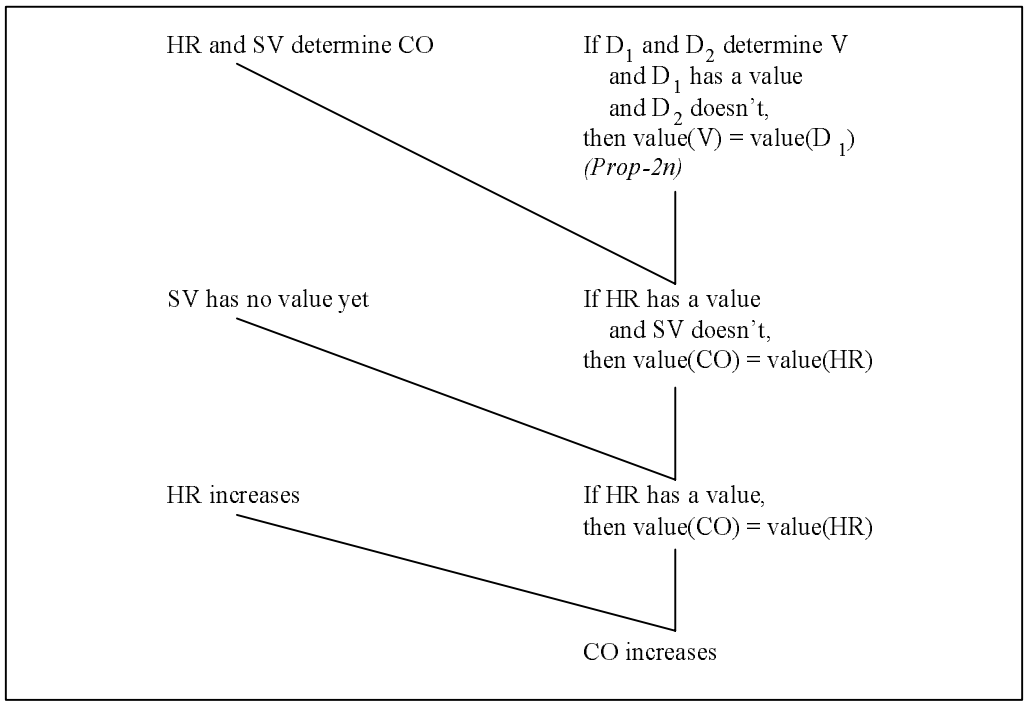

Figure 5.2: Derivation of the value of a non-neural variable

2) *T-teaches(P(x))*
This form is used when we want the proposition P(x) to be taught at this point even if it has been stated earlier in the conversation. This form occurs frequently in our dialogue schemata because many argumentative structures require information to be stated explicitly. (For example, there is a big difference between "I'm not going" and "I'm not going because it's raining," even if everyone knows that it's raining.)

3) *T-conveys(P(x))*
This form is used to state the proposition P(x).

4) *T-elicits(x, P(x))*
This form is used when the tutor wants to obtain some information from the student. The additional argument is used to identify the desired information.

These semantic forms are related as follows:

S-knows(P(x)) *if* P(x) is already known *or* *T-teaches(P(x))*
T-teaches(P(x)) *if* *T-elicits(x, P(x))* *or* *T-conveys(P(x))*

5.2.2 Domain-dependent semantic forms

Since every plan operator expresses a communicative act, the term *semantic form* can be used to describe any plan operator, but the term seems most appropriate for low-level schemata which do not interact with the student. Such schemata are used to provide further detail about the language to be generated. For example, the following schemata show two ways to implement the semantic form *introduce-stage*.

Introduce-stage (?stage):
    State-existence-of (errors)
    Point-at (?error1)

Introduce-stage (?stage):
    Review-predictions (student-predictions (?stage))

Together these two schemata can be used to generate all of the examples in Section 4.3.2. The first example below could be generated by the first decomposition above, while the remaining examples require the use of the second.

T: … There are some errors here. Let's start with this issue…
(K14:31)

T: … Now let's review some of your predictions…
(K32:46)

T: … Let's take a look at some of your predictions…
(K10:29)

T: … let's talk about your predictions…
(K27:50)

In Section 5.4.3 we will examine the lexical entries necessary to generate these examples.

We do not make a formal distinction between plan operators which interact with the student and those which don't because many operators can be implemented both ways. For example, *introduce-stage* could also be implemented by an operator which asked the student to identify the next stage.

5.2.3 Independence of semantic primitives and surface structure

The choice of semantic primitive does not constrain the form of the eventual surface structure. For example, *T-elicits(x, P(x))* does not require that the resulting text have the surface form of a question, but could be instantiated using any of the major sentence structures:

Interrogative: *(direct)* How is TPR controlled?
(indirect) Could you tell me how TPR is controlled?
Imperative: Please tell me how TPR is controlled.
Declarative: I'd like to know how TPR is controlled.

Similarly, *T-conveys(P(x))* need not be realized only by a declarative sentence:

Interrogative: Did you forget that TPR is neural?
Imperative: Remember that TPR is neural.
Declarative: TPR is neural.

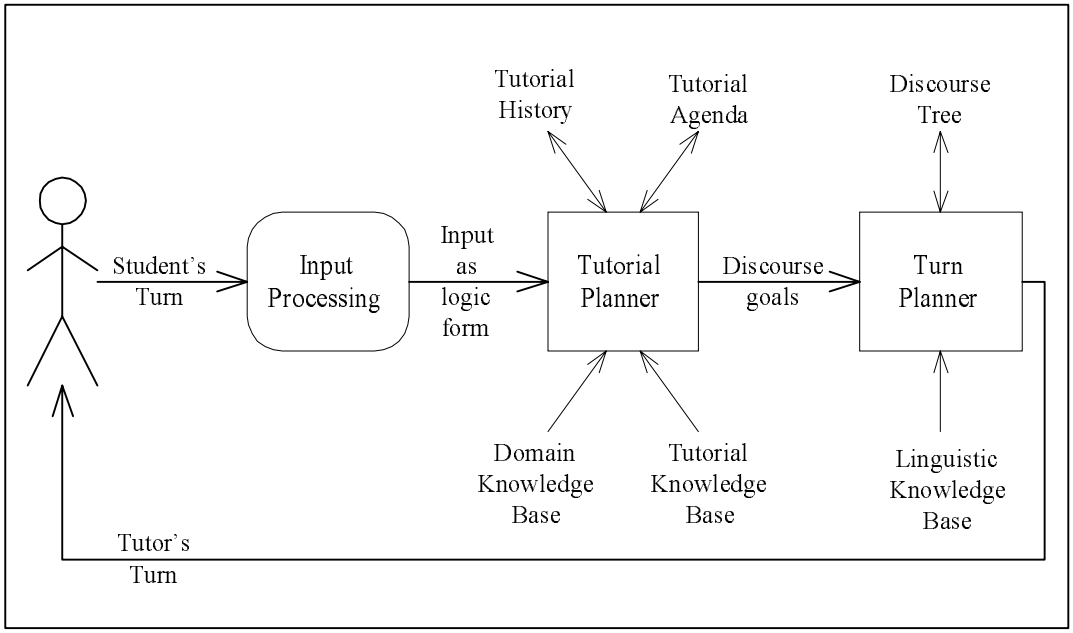

Figure 5.3: CIRCSIM-Tutor v. 3 from the point of view of text generation

5.3 Tutorial planning: Generation of semantic forms

5.3.1 Architecture of the discourse planner

Figure 5.3 shows an overview of the CIRCSIM-Tutor system from the point of view of text generation. CIRCSIM-Tutor can be viewed as a combination of two coroutines, one for input processing and one for output. The input processor accepts the student's input and converts it into one or more *logic forms*. It then passes control to TIPS, which generates a reply to be displayed on the student's terminal. The TIPS planner is a top-down global planner which can be considered a variant of a "classical" planner. The underlying model of TIPS is that responding to the student is a text planning problem, i.e. "what to say and how to say it." The top-level goal of the TIPS planner can be expressed as "generate text in which the tutor guides the student to the correct values of the variables."

TIPS contains two primary modules, the *tutorial planner*, which controls the conversation as a whole, and the *turn planner*, which assembles individual turns from the raw material provided by the tutorial planner. The tutorial planner does high-level discourse processing, i.e. it implements the larger-scale phenomena seen in the transcripts, including the dialogue structure and the high-level hierarchical structure (procedure, stage, variable and attempt) of the tutor's discourse. Thus the tutorial planner decides "what to say." For large-scale argumentative decisions, such as the type of argument to be used for each variable or when to drop one form of argument and try another, the tutorial planner also decides "how to say it." The turn planner decides "how to say it" within a turn, i.e. how to break the turn into sentences and the lexical and syntactic realization of each sentence. Since decisions such as pronoun use may require reference to previously generated turns, the turn planner must be able to access information about previous turns.

Although classical planners are often used for planning monologues, a planner cannot be used in the same way to plan a dialogue. A dialogue cannot be completely planned in advance because we cannot predict how the student will respond. Instead, the architecture required for dialogue planning is similar to that required for playing a two-person game such as chess. A chess program can plan many moves ahead, but it must be ready to change its plan based on its opponent's response. Similarly, the tutor can set up multi-turn plans, but it must be ready to change those plans if the student does not respond as predicted.

We plan until the next move has been determined, then execute that move and wait for the student's response. We store the tutorial agenda so that when the tutorial planner gains control again, we can resume where we left off. For efficiency, we do not expand nodes until they are needed.

Each correction mechanism is expressed as a *dialogue schema* which contains the raw material for achieving a communicative goal. A dialogue schema contains one or more turns, each of which comprises one or more subgoals. Each schema is represented as a plan operator. The prerequisites of the plan operators control the decision-making process. Some of the conditions tell the tutorial planner when the schema applies. Others tell the planner which responses on the part of the student are sufficient to continue the schema. The prerequisites are written in a declarative style, but they utilize a small number of executable modules, such as for sequencing the variables to be corrected. If the student gives a wrong or unexpected response, the planner will search for another schema, either an error recovery schema or an alternative method of tutoring.

The TIPS planner implements the planning process described in Section 5.1.1. A high-level plan built from operators such as those in Sections 4.3 and 4.4 chooses the sequence of variables to correct in each stage. In each step of this plan, we plan text for correcting one variable. Sections 4.6 and 4.7 describe major plans for correcting neural and non-neural variables respectively. Instructional planning, i.e. deciding which text plan to use next, is implemented via the prerequisites of the tutorial plan operators. Section 5.3.4 describes inputs which can be used in prerequisites. Curriculum planning could also be implemented through the use of plan operators and prerequisites.

The tutorial planner chooses a set of *primitive semantic forms* to express the content of the turn. The turn planner turns the semantic forms[1] into a cohesive piece of text which fits in the context of the previous turns. From the point of view of the tutorial planner, the turn planner is the plan execution module. The following chapter describes the

[1] Where there is no ambiguity, we often refer to primitive semantic forms simply as semantic forms. More generally, we use the term *semantic form* to refer to any plan operator, especially one which does not interact with the student. This type of plan operator is discussed in section 5.1.3.

work done by the turn planner to create a coherent paragraph from the set of semantic forms.

There are two primary reasons to accumulate a turn's worth of semantic forms before calling the turn planner. First is the fact that multiple semantic forms might contribute information to one sentence. Second is the issue of maintaining focus inside the turn. This requirement implies that we cannot choose the surface structure independently for each sentence.

We have chosen to use the LONGBOW system [Young 1994], a general-purpose discourse planner, described in Section 1.2.2, as the basis for the tutorial planner. Therefore we must build a mechanism to permit intermittent planning with the LONGBOW system.

5.3.2 Knowledge sources used in text generation

The tutorial planner maintains two knowledge sources and uses two others for reference:

- *Domain knowledge base*
  The domain knowledge base contains the physiological facts and rules which CIRCSIM-Tutor deals with. It contains domain-specific non-discourse based knowledge. In addition to pure domain knowledge, i.e. the facts and rules of cardiovascular physiology, it contains domain-specific pedagogical information, such as the preferred order for correcting variables.

- *Tutorial planning knowledge base*
  The tutorial planning knowledge base or *plan library* contains plan operators for the tutorial planner. It contains methods for teaching each concept and information about when to use each method. This information is domain-specific, discourse-based pedagogical knowledge.

- *Tutorial planning agenda*
  The tutorial planning agenda contains the currently open goals of the tutorial planner. It is an agenda, not a stack, because the tutor may need to change the top goals in order to respond better to the student.

- *Tutorial planning history*
  The tutorial planning history contains historical information about the tutorial planning process, i.e. which goals the tutor has selected, how they were implemented and which input from the student triggered their use.

The turn planner uses one knowledge source for its own use and one for reference:

- *Linguistic knowledge base*
  The linguistic knowledge base contains syntactic and lexical information necessary for realizing primitive syntactic forms as text.

- *Discourse tree*
  The discourse tree contains a representation of the generated text. It is used to ensure cohesion between the turn being generated and previous turns. The creation and use of the discourse tree are discussed further in Section 5.4.7.

5.3.3 Goals of the input processing phase

The goal of the input processing phase is to produce a logic form representing the student's input. Input processing has four major goals:

- Spelling correction
- Parsing
- Semantic interpretation
- Identifying the goal which the student is addressing

Since the input processor and the text generator communicate only via logic forms, the choice of methods for parsing and semantic interpretation and the degree to which these functions are performed in series or in parallel do not affect the text generator. Furthermore, the input processor does not need to understand everything the student says

or understand everything to the same level of detail. Note that we do not have to respond to every aspect of the student's input. If the student argues that "X because Y," we may have a response which is appropriate whenever the student says "X," whatever the reason. As long as the logic form contains enough information for the tutorial planner to choose an appropriate response, even a full parse may not be necessary.

Most existing software assumes that the human-machine interaction is structured by one party or the other. Since conversations with CIRCSIM-Tutor are structured by the tutor, we can assume that the student, as a cooperative conversation partner, does not have an independent plan. Therefore, instead of a formal plan recognition phase, it is sufficient to identify the goal which the student is replying to. In most cases, we can assume that the student's reply refers to the question most recently asked. Checking the tutorial agenda is a useful way to identify situations such as the following, where the student's reply refers to a higher-level open goal on the agenda:

(1) T: What is the value of CVP?
S: I don't know.
T: OK, what are the determinants of CVP?
S: Oh, I know! CVP increases…

Although identification of the student's goal could be accomplished by the tutorial planner, considering it part of the input understanding phase allows the input processor to use information about the student's goals to disambiguate situations in the parsing and semantic interpretation tasks which would otherwise require additional parsing knowledge.

Once the student's input has been interpreted, control passes to the tutorial planner.

5.3.4 Knowledge used for making instructional planning decisions

The tutorial planner obtains control once per turn. When it obtains control, it searches a library of plan operators to determine possible responses to the student's input. The plan operators decompose communicative goals into smaller goals until primitive semantic forms, which are undecomposable, are obtained.

The tutorial planner can use many types of available input to decide which plan operator to invoke. Here are some possibilities:

- *Student's most recent statement*
  - Content of the student's response.
  - Classification of the student's response as described in Section 4.2.2.
- *Domain knowledge*
  Is the variable neural?
- *Pedagogical history (non-discourse related)*
  - How accurate are the student's original responses in the prediction table?
  - Is the student responding to the tutor's hints?
- *Pedagogical history (discourse-related)*
  - How many attempts have been made to tutor this variable?
  - Are we currently in the middle of a multi-turn plan?
- *Discourse history*
  Although syntactical and lexical information is available and might occasionally be useful for ensuring cohesion, this information is not in general relevant to the high-level decision-making being discussed here.
- *Tutor's personal style*
  This factor is primarily applicable to the human tutors, as we do not attempt to formalize this issue for machine-generated text.
- *Random choice*
  We do not always have enough pedagogical knowledge to make a principled distinction between similar forms. Thus random choice is a useful aid to encourage the use of a variety of schemata.

We maintain a history of our interactions with the student, containing information about the classification of the student's input and the goals chosen in response, for use by the tutorial planner in future planning steps. This history can be used in order to ensure that we don't unnecessarily repeat plans from our plan library.

5.3.5 Generating material for a turn: the replanning operators

Sections 4.2.2 and 4.5 describe the structure and content of the turns emitted by our expert tutors. We recall that a turn has the following sections:

Turn:
- Response to student's previous statement
  - Acknowledgment of student's statement (optional)
  - Content-oriented reply (optional)
- New material (optional)

According to the protocol described in Section 5.3.7 below, the turn must end in a question or request for the student.

The first step in tutorial planning is to decide whether to issue an explicit acknowledgment. If the potential acknowledgment is negative, this decision must be made in conjunction with a decision about issuing an explicit rebuttal. If an acknowledgment is to be issued, the tutor chooses the direction and strength for the acknowledgment and formulates a semantic form. This semantic form will be the first entry in the buffer to be passed to the turn planner for this turn.

One of the factors that makes much computer-generated text sound artificial is that each idea is fully spelled out, whereas human beings are used to using their reasoning powers to understand a less explicit form of language. Therefore we want to model our acknowledgments on human tutor behavior, although much work remains to be done on classifying exactly when and how to acknowledge. For example, when the student makes

an error, we know that human tutors give an explicit negative acknowledgment only about 25% of the time, whereas CIRCSIM-Tutor v. 2 always does. It has been suggested that the tutor is more likely to choose an explicit acknowledgment if the student's response contained a hedge (e.g. *maybe* or *I think*).

Some examples of discourse knowledge which CIRCSIM-Tutor v. 3 could use include:

- If the student gives values for two variables and only one of them is correct, it is often not necessary to give an explicit negative acknowledgment. If we say "the value of V_1 is correct," the student will infer that the value of V_2 is wrong.
- If we ask the student which variable is affected next and we get a wrong answer, it is not necessary to give an explicit negative acknowledgment. Our not following up on that variable is sufficient indication to the student that an inappropriate variable has been chosen.

Once the acknowledgment has been determined, we begin to plan the content-oriented part of the response. If the student has given the expected response, usually the correct answer, we take the next subgoal from the top of the agenda and begin to plan it. The planner runs, expanding the node at the top of the agenda during each cycle. When a primitive semantic form is reached, it is added to the turn buffer. Since a turn ends when a response from the student is required, the first primitive semantic form to request a response from the student ends the turn. At that point control is passed to the turn planner. The agenda is saved for the tutorial planner when it regains control.

If the student did not give the expected response, we can recover in one of three ways. First, we can retry the tutor's last goal with a new instantiation. Second, we can add a goal to the top of the agenda, an operation known as *operator preposing* [Wilkins 1988]. If the student's answer is pedagogically close to the desired answer, we

may have a schema to help the student take the next step. If it is linguistically close, we would like to give the student a better way to state the answer. If we recognize the student's wrong answer as indicative of a particular confusion, we might have a correction schema to respond to it.

Alternatively, we can switch to a schema with more potential. This operation is known as *operator replacement*. If we don't understand the student's response or there is no schema available to respond specifically to it, we can drop the current schema and choose a new one. Our human tutors often use a cue phrase such as "Let me try again" to mark a new approach. Since schemata can be nested, the agenda contains all higher-level goals in process as well as the younger siblings of each of those nodes. The latter represent the future steps of each schema currently in play. If we are not currently processing the last subgoal of the current schema, its remaining subgoals of the current schema must be removed from the agenda before a new schema is chosen.

Although we can remove undesired subgoals of a failing schema from the agenda, we do not want to lose track of turns already uttered in service of the already-satisfied subgoals. For example, suppose we wanted to terminate a four-subgoal plan after the second subgoal. As we can't "unsay" a turn, the already-generated text would still be part of the conversation and would still be relevant when attempting to increase the cohesion of later turns. For this reason we must maintain a data structure for the developing conversation separate from the tutorial planning agenda. This *discourse tree* is described in Section 5.4.7.

Since we accumulate semantic forms until we reach a form requiring a response from the student, the semantic forms comprising a turn may (and probably do) originate in different schemata. A semantic form may be derived from the acknowledgment, the

content-oriented reply or the new material. Within the new material, since planning goals may be nested, semantic forms may be generated from multiple subgoals at several levels.

5.3.6 Replying to student initiatives

Although we do not handle the most general type of student initiative, we can handle most of the initiatives we expect students to use. Examples of student initiatives which we can handle include the following:

- Request for an explanation of something the tutor has said
- Request for a definition
- General request for help

Since users do not expect the computer to be a full conversation partner, we expect the majority of student initiatives which CIRCSIM-Tutor will encounter to be included in these categories. The TIPS planner handles these initiatives just as it would handle any other unexpected response from a student. Depending on which matching plan operators it finds in the plan library, it can respond in one of the following ways:

- Put the student's request on the top of the agenda
- Save the student's request for future consideration
- Add a discourse goal to explain to the student that the request must be deferred

5.3.7 Pragmatic restrictions on turn design

CIRCSIM-Tutor ends each turn with an explicit request for information from the student although our human tutors occasionally leave it implicit. There are several reasons for this decision. First, it improves our chances of getting the kind of answer we want. Second, the use of explicit questions reduces the chance that the student will say something which the input processor cannot understand. A third reason for the use of explicit questions is philosophical. Since the program can never really know whether the

student understands the material, an operational definition, such as whether the student can correctly answer questions, is extremely useful. Finally, people expect to deal with a computer in a question-and-answer mode. Turn-taking rules work in person-to-person conversation because we are socialized to understand and use them [Sacks, Schegloff & Jefferson 1974]. But people have different expectations from a computer [Dahlbäck & Jönsson 1989].

For example, the human tutor might say "First we need to know how TPR is controlled," knowing that the student will respond cooperatively. CIRCSIM-Tutor can achieve the same communicative intent with short-answer questions such as "By what mechanism is TPR controlled?" or "What controls TPR?".

A second pragmatic restriction on turn design is to prefer short-answer questions such as the ones suggested above to open-ended questions such as "How is TPR controlled?". In general, we do not want to ask questions to which we might not understand the answer. Even if the input processor could understand the reply to a particular open-ended question, the available domain knowledge is not deep enough to make use of an arbitrary reply for pedagogical planning purposes.

These two pragmatic restrictions do not reduce the range of topics we can discuss. However, since the use of open-ended diagnostic questions is restricted, the tutor must use other avenues to identify the source of the student's confusion. Alternatively, the tutor can use heuristics for choosing a response which do not require explicit identification of the student's problem. We generally choose the second approach.

5.3.8 Syntax and semantics of the tutorial planning operators

In this section we give a formal definition of the tutorial plan operators. Plan operators in CIRCSIM-Tutor have four fields:

- Name
- Effects
- Constraints
- Decomposition

The intended interpretation of the plan operator is as follows:

- The *name* field identifies the communicative act which the rule is intended to implement.

- The *effects* field identifies any state changes which can be assumed to be true as a result of the operator.

- The *constraints* field refers to *a priori* preconditions which must be true before the operator will be considered. Constraints are also called *filters*.

- The *decomposition* field describes how to implement the communicative act represented by the *name*, or equivalently how to obtain the state described by the *effects*, in terms of lower-level operators or primitive semantic forms. If it contains more than one entry, the subgoals must be satisfied sequentially. We also have a notation which means "as many times as possible" so that indefinite repetition, such as iterating over each incorrect variable, can be expressed efficiently.

A plan operator may be selected for use if its name (including the arguments) or its effects can be matched against the current entry in the decomposition of the plan operator currently being expanded. Arguments are matched using full unification.

The planning literature contains three kinds of preconditions for plan operators:

- Preconditions which must be *a priori* true; the planner will not try to make them true. For example, an ambulance will take you to the hospital, but it does not make sense to have an accident to get a ride to the hospital. In addition to ruling out inappropriate behavior, this type of constraint is useful for reducing the search space.

- Preconditions which the planner can try to make true. If they are already true, then no action need be taken. This is the most common case in planning other than text planning.

- Preconditions which the planner must instantiate even if they are already true. This category is used where the planner must generate text to make a coherent argument even if it has already been said or the student already knows it. This type of precondition is often used to generate the premises of an argument when they are realized as subordinate clauses. For example, a speaker might say, "Since it is raining, we cannot go swimming," even though the hearer already knows it is raining.

The TIPS planner handles all three types of preconditions. The first type is located in the *constraints* field of the plan operator. The second and third types of preconditions both occur in the *decomposition* field. Since plan operators may match on either the operator name or the effects, we adopt the convention that entries which contain the name of an operator must be implemented, while entries stated in terms of a desired effect are skipped if the effect is already true. The intention is that if the plan operator calls for a specific communicative act on the part of the tutor, the tutor must perform that act. If the plan operator simply requests that a state be true, no action need be taken if the state is already true.

Plan operators are named from a third-party point of view, e.g. *T-elicits* to describe an action taken by the tutor and *S-knows* to describe the student's desired state. For plan operators which could be implemented either way, we use a passive form such as *variable-is-introduced* or a non-specific form such as *introduce-variable*. Most of our operators are defined from the tutor's point of view because we know the tutor's goals.[2] From a philosophical point of view, we can never be sure what the student knows or believes. We prefer to define our plan operators using things which we have direct knowledge of, such as the dialogue thus far and the tutor's goals, rather than things we

[2] In Chapter 4, we name all of the schemata from the tutor's point of view because the analysis of the transcripts was done from the tutor's point of view. In that chapter, we use forms such as *ensure-student-knows* to describe the desired state of the student from the teacher's point of view.

can only guess at. Furthermore, describing a goal from the tutor's point of view is sometimes required when the text to be generated is part of a larger tutorial pattern. If we cannot control how the different parts of the pattern are realized, we may not be able to ensure that they have the desired relation to one another.

Since subgoals are recursively evaluated, the fact that a schema is implemented as a single plan operator does not determine how many turns it will eventually generate. Section 4.6 gives examples showing that one schema can generate dialogues of varying length. Additional subgoals can be generated if a subgoals invokes a nested schema or if the student responds incorrectly to a question inside a schema.

Adjunct schemata, such as schemata for generating rebuttals or restatements of the student's statements often generate a single turn or a part of a turn. Other schemata which often generate a single turn include *give-definition*, *give-rule* and *give-answer*. Hints are usually generated by longer schemata from which only one subgoal is chosen to generate text. Other plans, such as interactive explanations, take place over several turns as long as the student gives "cooperative" (in the Gricean sense) responses.

5.4 Turn planning: Realization of semantic forms as surface structure

5.4.1 Functions of the turn planner

The turn planner is responsible for assembling semantic forms generated by the tutorial planner into a coherent turn which fits into the evolving conversation. Except for the fact that turns must fit coherently into the evolving conversation, any general-purpose paragraph planner could potentially be used for the turn planning function. In this section we will sketch the functions of the turn planner and show some of the goals which it needs to accomplish, but we will not propose a detailed implementation.

Turn planning includes the following functions:

- Organization of semantic forms into sentences
- Lexical insertion
- Text realization
- Post-linearization editing
- Ensuring intra-turn coherence
- Ensuring inter-turn coherence

The turn planner starts by choosing which semantic forms will be realized in each sentence to be emitted, creating an intermediate form known as the *deep syntactic form*. The use of an intermediate form allows many kinds of transformations. Thus our ability to generate coherent text is not limited by the knowledge representation. The use of an intermediate form also simplifies the potential addition of the ability to do multi-language generation since only the lexicon and grammar refer to a specific natural language.

The *lexical insertion* process converts each deep syntactic form to a *surface syntactic form*. Lexical insertion includes the following functions:

- Choosing a main verb for a process
- Determining the syntax required by that verb
- Choosing natural language representations for the arguments and modifiers of the verb

The *text realization* process creates the surface text. This step includes making syntactic and morphological choices, including decisions about morphological forms, agreement, word order, and the selection of closed class forms. We plan to use the FUF package [Elhadad 1993] for this step. *Post-linearization editing* refers to syntactic choices which cannot be made until the word order of the sentence is known, such as decisions about when to use pronouns.

Intra-turn coherence means that the sentences of a turn need to flow together as a paragraph. To ensure this requires coordination between the syntactic choices made for

the sentences in a turn in order to maintain a smooth flow from each topic to the next. Other factors such as pronoun usage also influence intra-turn coherence. *Inter-turn coherence* means that the turn fits into the evolving conversation.

5.4.2 Accumulation of semantic forms into sentences

Even within a turn, the use of schemata or their equivalent is required for a system with the desired speed and coverage of CIRCSIM-Tutor. To generate text via intention-based planning is impractical not only because of the volume of computation required [Appelt 1985; Moore 1995], but because the rhetorical knowledge necessary for combining semantic forms into sentences is not available for the large number of cases which we expect CIRCSIM-Tutor to handle.

Since our domain logic consists mostly of simple causal chains, in many cases we can generate reasonable text by realizing each semantic form as a sentence. However, there are cases where a more natural result or a useful variation can be obtained by combining multiple semantic forms into one sentence. The human tutors often combine an acknowledgment with a content-based reply, as in the following examples.

(2) T: Right, SV goes down…

(K14:55)

(3) T: … What is the stimulus for the baroreceptor reflex?
S: A change in the pressure in the system.
→ T: Right, a change in MAP…

(K36:78–80)

In the following example, an acknowledgment is combined with new material, namely a semantic form derived from the current correction schema.

(4) T: … And what will follow from this change in SV?
S: CO will decrease.
→ T: Right, so what happens to RAP in DR?
(K37:218–220)

In the following example, a partial acknowledgment is built from multiple semantic forms. In the student's answer, CC is correct but HR is wrong.

(5) S: CC would also decrease and so would the HR.
T: Right, except remember in this guy HR is determined by the artificial pacemaker so the reflexes can't affect it…
(K2:39–40)

In the following example, the first sentence includes an instance of *start-new-attempt* and the first semantic form from *show-contradiction*. The *show-contradiction* schema continues with the next sentence.

(6) T: … But let's go back to your definition of DR, and remember that you predicted that TPR would increase. Do you want to rethink this?
(K37:52)

5.4.3 Lexical insertion

Lexical insertion refers to the process of choosing a lexical item for each concept represented in the deep syntactic form.

In Section 5.2.2 we showed two potential implementations for *introduce-stage*:

Introduce-stage (?stage):
State-existence-of (errors)
Point-at (?error1)

Introduce-stage (?stage):
Review-predictions (student-predictions (?stage))

To realize each of these options as surface text, lexical entries would be necessary for the following verbs (the term 'verb' here includes phrasal verbs):

- *Review*
- *Take a look at*
- *Talk about*

Although each of these verbs has a slightly different semantic range, the distinctions between them are not relevant to CIRCSIM-Tutor. Since each lexical entry for a verb points to the appropriate case information for its arguments, all that remains is to instantiate the arguments and modifiers and build the surface form of the sentence.

If the speaker were an employee wishing to show an error to the boss instead of a tutor wishing to show an error to a student, we might want to generate text with the semantic content of *review* but in a different register. For example, we might want to generate a sentence such as the following:

(7) I'm having trouble understanding some of your predictions.

We could implement the ability to generate this sentence either by adding additional lexical insertion rules for an existing semantic form or by adding a new semantic form. Although there is a conceptual difference between these options, there is no practical difference with regard to the generation of the sentence. In the long run, one of the choices might lead to a more logical set of semantic primitives.

Thus we take a pragmatic view toward the set of semantic primitives. For example, here are some ways to express the fact that an increase in HR causes an increase in CO:

(8) An increase in HR causes an increase in CO.
When HR increases, CO increases.
If HR increases, CO increases.
HR increasing causes CO to increase.
CO increased because HR increased.
HR determines CO.

Additional information on the range of sentences to be generated and the context in which they will appear is required in order to decide how many of these must be differentiated. There are two reasons why it is useful to have a number of realizations available for one semantic form. In some contexts, the topic/focus rules discussed in the following section restrict which of these forms can be used in a given context. Even if those rules do not apply, lexical equivalence is one way to increase variety in our generated texts.

Reiter [1991] points out the importance of making principled decisions in the lexical insertion process. However, there are cases where random selection appears to be sufficient. For example, it is often claimed that human tutors tend to use the word family previously used by the student when choosing, for example, between the word families *go up/go down* and *increase/decrease*. The transcripts do not bear this out. Of course, if desired, this feature is easy to implement by having the input understander set a flag indicating that one of the target verbs has been encountered. The discourse tree (Section 5.4.7) can be used to discover whether we are still in a discourse segment where one of the target verbs has been used.

Backtracking with respect to lexical insertion is an open issue. If the tutorial planner can request a retry with a new lexical choice, then there must be a way to signal that no additional choices are available.

5.4.4 Intra-turn coherence

We generate text a turn at a time in order to ensure that the input necessary to create coherent turns is available. Although coherence is one of the most complex aspects of text generation, studying the output of v. 2 leads us to believe that it is only necessary to consider a few of the most common phenomena in order to generate reasonable text for v. 3. In this section we briefly consider three of the most frequently occurring phenomena:

topic/focus rules, pronoun usage, and the use of discourse markers.

Consider the following sentences:

(9) TPR is a neurally controlled variable.
TPR is neurally controlled.
TPR is neural.
TPR is controlled by the nervous system.
The nervous system controls TPR.

In some cases, sentences such as the following might be added to the list.

(10) Did you forget that TPR is neural?
Remember that TPR is neural.

Although all of the sentences in (9) could be generated by the same semantic form, they are not equivalent in all contexts because topic/focus relationships dictate which of the previously mentioned nouns in a paragraph can be the subject of the succeeding sentence. Additionally, depending on the previous sentences in the turn, it might be necessary to replace one of the nouns by a pronoun in order to obtain a natural-sounding paragraph.

There are several approaches available to help ensure intra-turn coherence. If a turn is built from a pre-planned pattern or contains only one semantic form other than the acknowledgment, then the issue does not arise. However, in other cases it may be useful to implement rules such as those discussed by McKeown [1985, ch. 3] or Hovy and McCoy [1989].

Finally, discourse markers play an important role in creating coherence. Since discourse markers such as *so* or *and* are usually associated with a schema entry, they can contribute either to intra-turn or to inter-turn coherence, depending on how the schema is instantiated. The following section contains some examples of the use of discourse markers.

5.4.5 Inter-turn coherence

The use of a global planner with schemata for generating text contributes greatly to inter-turn coherence because coherence of the argument is one of the key factors in assuring inter-turn coherence.

Second, the use of discourse markers [Schiffrin 1987] is a small element which greatly increases the readability of the resulting text. Consider how each of the following examples would sound if the *so* were omitted from the final sentence.

(11) T: … Can you tell me how TPR is controlled?
S: Autonomic nervous system.
→ T: Yes. And the predictions that you are making are for the period before any neural changes take place. So what about TPR?
(K10:29–31)

(12) T: … What are the determinants of MAP?
S: CO and TPR.
→ T: Correct. And you have predicted CO increases and TPR increases. So how can you say MAP decreases?
(K32:226–228)

Chapter 4 contains many additional examples of the use of discourse markers.

These two factors suffice for the dialogues currently proposed for CIRCSIM-Tutor v. 3. A third factor which has been mentioned is the ability to generate syntactic parallelism. For example, surface forms which belong to the same interactive explanation might all begin with "And …", "So …", or a more complex grammatical structure. This feature can be implemented by having either the parent goal or the first sibling set a flag which will be checked in the prerequisites of later plan operators. This situation is similar to the selection of parallel lexical items described in Section 5.4.3.

5.4.6 Text realization and post-linearization editing

We plan to use the FUF package [Elhadad 1993] for surface realization. Several aspects of FUF and the accompanying SURGE grammar seem especially well-suited to CIRCSIM-Tutor:

- FUF forms have a coherent ontology.
- FUF forms have the power to say most things we need to generate, such as compound and complex sentences and subordinate time clauses.
- FUF forms are extensible.

Some features of the surface syntax, such as choosing to use a pronoun or to combine two clauses into a single clause with a plural subject, cannot be determined until the surface word order is known. Some basic rules on the use of pronouns are provided by Fox [1987]. We expect that post-linearization editing to be minimal in the initial CIRCSIM-Tutor implementation.

5.4.7 Building the discourse tree

The turn planner adds the form to be uttered to a discourse tree based on Conversation Analysis principles (see Section 1.3.4). Some examples of the discourse tree are shown in Section 5.5.3. As mentioned in Section 5.3.5, when a pedagogical schema is dropped, turns which have already been uttered must still be represented in the discourse tree. In addition, we would like to use the discourse tree to make decisions on items such as pronoun usage. Thus we do not want the structure of the discourse tree to be restricted by the need to be consistent with the pedagogical goals. For example, supposes the tutor challenges a student idea by saying, "In that case we could deduce …" Although that statement is obviously the response to a student utterance, depending on the content of the statement, it might make more sense to consider it the beginning of a new exchange. When

the student gives an incorrect answer and the tutor shifts course, it is not always obvious when a new discourse topic has been started. Since the purpose of the discourse tree is to provide information for the generation of later turns, this issue can remain unresolved until we are ready to make decisions based on this information.

Sometimes we need to wait until the student's response is received before we know how to categorize the tutor's previous statement. For example, if the student gives a serious answer to a pseudo-diagnostic question, we are more likely to want to treat the question as part a *T-elicit* exchange than if the student realizes our intention.

5.5 Examples of the dynamic behavior of the discourse planner

5.5.1 Example showing options in the planning process

With plan operators based on the schemata in Chapter 4, we can use the planner described in this chapter to produce multiple ways to tutor any concept in the domain of CIRCSIM-Tutor. In this section we demonstrate this proposition for two cases, a non-neural variable and a neural variable.

Consider the broken pacemaker problem in Section 2.4.2. Suppose the student gives the correct value for heart rate but not for cardiac output. There are a large number of ways we can proceed in order to correct cardiac output. Here is one instantiation of each method.

1) Teaching via determinants (Section 4.7.1)
   T: What are the determinants of CO?
   S: SV and HR.
   T: Which is the major determinant?
   S: HR.
   T: What is the value of HR?
   S: It increases.
   T: So what is the value of CO?

2) Teaching via main determinants (Section 4.7.1)
   T: What determines CO?
   S: HR.
   T: And what is the relation between HR and CO?
   S: Direct.
   T: So what is the correct value of CO?
3) Building on the previous step (Section 4.7.2)
   S: HR increases.
   T: And what other variables does that affect?
4) Pointing out a contradiction (Section 4.5.4)
   T: You said HR increases but CO decreases. How can that be?
5) Invoking an equation
   T: Can you give me an equation relating SV, HR and CO?
6) Invoking a definition (Section 4.9.1)
   T: Can you give me the definition of CO?
7) Switching to deeper concept map (Section 4.10.1)
   T: CO is the volume of blood pumped by the ventricle in one minute…

Each method can be instantiated using varying degrees of interactivity. Here are two snippets of dialogue, both generated with *show-contradiction*, but with varying degrees of interactivity.

T: … What are the determinants of CO?
S: HR and SV.
T: Correct. And you have predicted HR increases and CO increases. So how can you say SV decreases?

T: HR and SV determine CO. You predicted that both HR and CO would increase. So how can CO decrease?

Within the chosen plan, the basic functions of asking for information and giving the student information can be implemented in a variety of ways. Here are some examples.

- *Eliciting information*
  What is the value of HR?
  Can you tell me the value of HR?
  Please tell me the value of HR.
  I'd like to know the value of HR.
- *Conveying information*
  Did you forget that HR increases?
  Remember that HR increases.
  HR increases.
  But SV increases.

Finally, here is a small sample of the variety of lexical and syntactic alternatives available.

- *Choosing lexical items*
  HR increases.
  HR goes up.
  HR rises.
- *Choosing syntactic forms*
  HR increases.
  There's an increase in HR.
  Because of the increase in HR, …

In summary, combinatorial explosion is a welcome feature here, as it enables us to generate a large number of dialogues at modest cost.

We now present a similar set of options for the correction of a neural variable. There is only one basic schema for correcting the first incorrect neural variable. Here are several ways to choose different levels of interactivity for it.

- Interactive explanation
  T: How is HR controlled?
  S: Nervous system.
  T: Right. And we're talking about what happens before there are any neural changes. Now what do you say about HR?
- Explanation with follow-up question
  T: HR is a neurally controlled variable. But DR is the period before there are any neural changes. So what value would you assign to HR in DR?

- Hint
  T: Remember that we're talking about what happens before there are any neural changes. Now what do you say about HR?

Here are some ways to implement individual plan steps which are useful in the neural domain.

- *Eliciting information*
  How is HR controlled?
  Could you tell me how HR is controlled?
  Please tell me how HR is controlled.
  I'd like to know how HR is controlled.
- *Conveying information*
  Did you forget that HR is neural?
  Remember that HR is neural.
  HR is neural.

Finally, here are some syntactic and lexical choices which are useful in the neural domain.

- N + *to be* + adjective
  HR is neural.
  HR is neurally controlled.
- Verb *to control* + arguments
  HR is controlled by the nervous system.
  The nervous system controls HR.
- N + *to be* + N
  HR is a neurally controlled variable.
  HR is a neural variable.

The actual number of dialogues which CIRCSIM-Tutor can generate is greater than predicted by combinatorial explosion alone because of the way it can switch schemata in response to student errors. In the following example, the tutor starts with a short form of *show-contradiction*, then switches to a longer form of *show-contradiction*, then gives the student part of the answer and switches to building on the previous step.

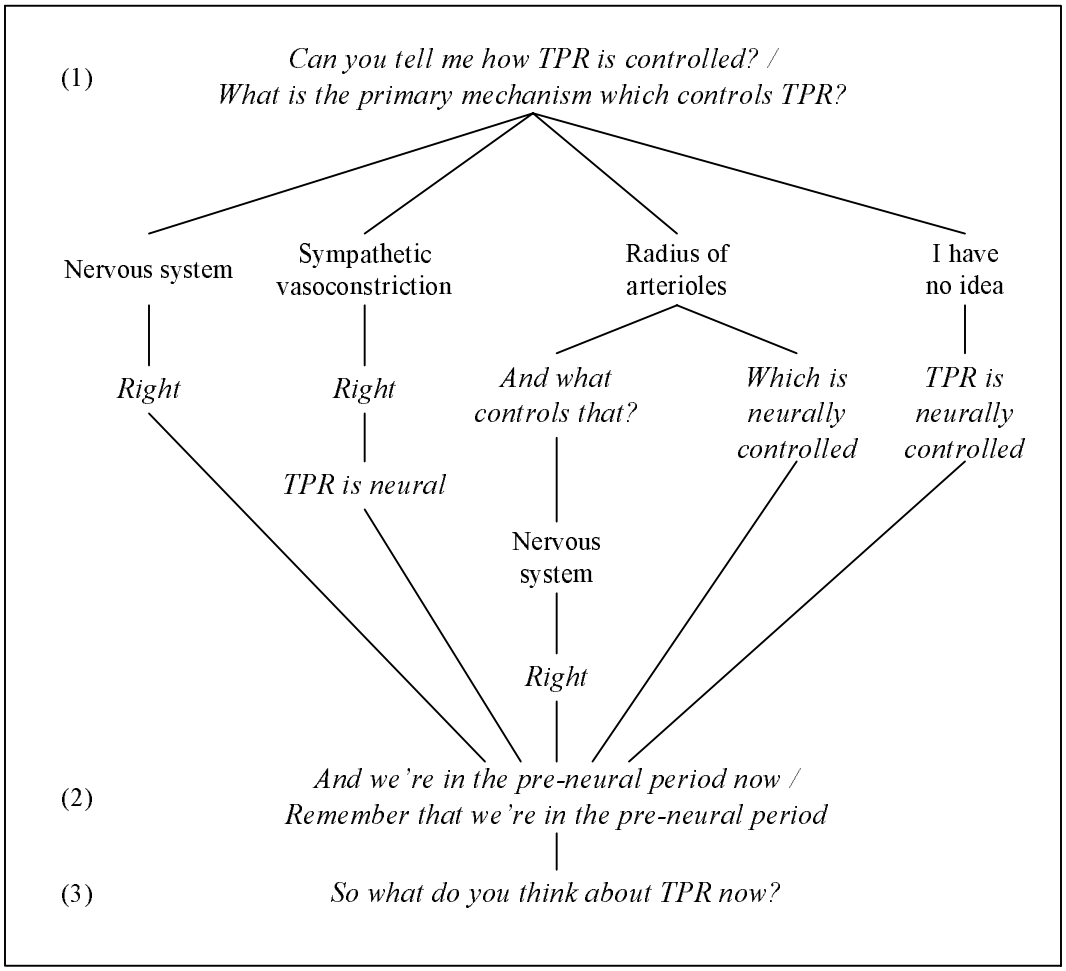

Figure 5.4: Dialogues which CIRCSIM-Tutor can generate

(13) T: You said HR increases and CO decreases. Is that possible?
S: Yes.
T: Let's try again. What is the value of HR?
S: It increases.
T: And what is the value of SV?
S: I don't know.
T: Well, at this point, SV is unchanged. Now if HR increases and SV is unchanged, what happens to CO?

The following two sections give detailed examples showing the use of adjunct schemata and the replanning algorithm, respectively.

5.5.2 Example showing dialogues which can be generated

Figure 5.4 shows in condensed form a set of dialogues which our system can generate from the *correct-neural* schema of Section 4.6.1:

Correct-neural (?v):
PQ: ?v is neural
?v is not the primary variable
S-knows (has-mechanism(?v, neural))
S-knows (current-stage(DR))
S-knows (has-value(?v, DR, no-change))

Each path through the diagram shows the effect of a different implementation of the first subgoal. Each item in italics represents text which could be generated by one semantic form. For the sake of variety, we try to provide multiple possible realizations for each semantic form. In a few cases we have shown two possible realizations separated by a slash. The numbers correspond to the sequence of the subgoals in the schema above. The student's responses are shown in roman type. In the leftmost path, the student gives the desired answer immediately. In the second path, the student gives an answer which is true but does not use the tutor's desired language. The tutor adds a goal to correct the student's language before continuing with the schema. In the third branch, the student gives an answer which is on the path toward the correct answer. The tutor helps the student toward the correct answer, using *T-elicits* and *T-conveys* respectively in the two sub-branches to implement the correction schema. In the final branch, the tutor uses the *give-answer* schema to satisfy the first subgoal.

By varying the instantiation of the other subgoals of this schema, as well as utilizing the option to switch to a different schema, we can generate a rich variety of dialogues for teaching this concept.

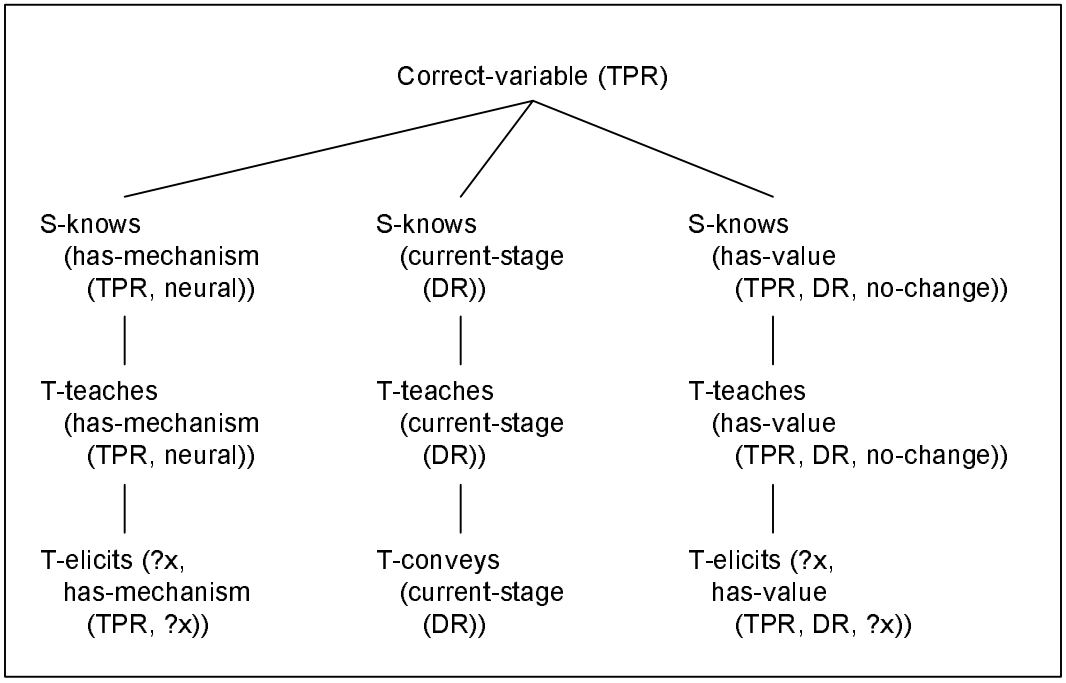

Figure 5.5: Conceptual tutoring agenda

5.5.3 Example showing the replanning algorithm

This example demonstrates the operation of the replanning operators discussed in Section 5.3.5. Additionally, it shows discourse trees which could have been generated from the plan trees shown. Again, we use the *correct-neural* schema introduced in Section 4.6.1.

Correct-neural (?v):
PQ: ?v is neural
?v is not the primary variable
S-knows (has-mechanism(?v, neural))
S-knows (current-stage(DR))
S-knows (has-value(?v, DR, no-change))

Since our planning algorithm only expands one subgoal at a time, a full goal tree never appears at a detailed level. To make the example easier to follow, Figure 5.5 shows what

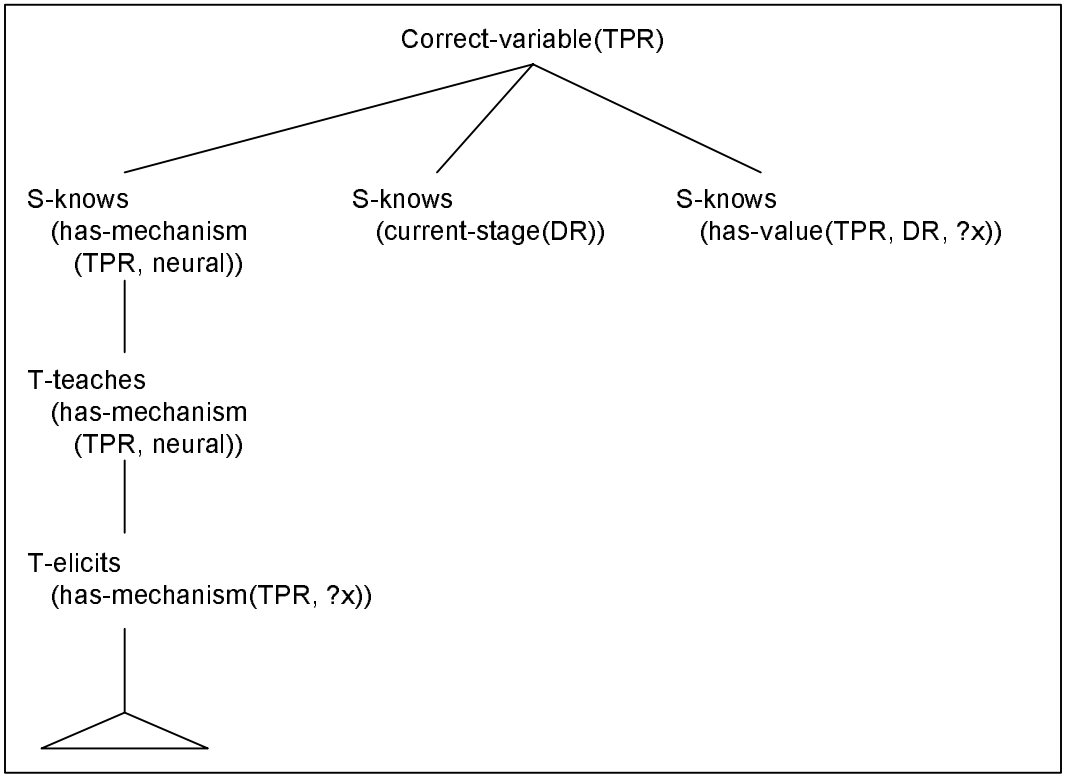

Figure 5.6: Initial tutoring agenda

such a tree would look like. The tutoring goals actually on the agenda after two plan steps are shown in Figure 5.6.

The following text, which is derived from one of our live tutoring sessions, could be generated from this plan.

(14) 1 T: How is TPR controlled?
2 S: Nervous system.
3 T: Right. And we're talking about what happens before there are any neural changes. Now what do you say about TPR?
4 S: It doesn't change.
5 T: Right.

(K11: 49–53, slightly abridged)

Section 5.4.7 describes the creation of the discourse tree. The discourse tree corresponding to this tutoring plan is shown in Figure 5.7. The Conversation Analysis terminology is defined in Section 1.3.4. Note that turns are not coterminous with exchanges; in fact, a turn often ends one exchange and starts another.

To simplify the diagram, the acknowledgment ("Right.") is turn 3 has been omitted from Figure 5.7. In general, an acknowledgment closes a *T-elicit* exchange. Yet it does not seem altogether natural to separate the acknowledgment from the following sentence with which it is intimately related.

Now let us assume that the student does not give the correct answer in turn 2 above. Here is the text we would like to generate:

(15)
1 T: How is TPR controlled?
2 S: Autoregulation.
3 T: No, that's too slow to be relevant here. What is the primary mechanism of control of TPR?
4 S: The radius of the arterioles.
5 T: And what is the primary mechanism by which the arteriolar radius is controlled?
6 S: I don't know.
7 T: The sympathetic nervous system controls TPR. And we're dealing with the period before any change in nervous activity occurs. Now what do you say about TPR?
8 S: It doesn't change.
9 T: Right.

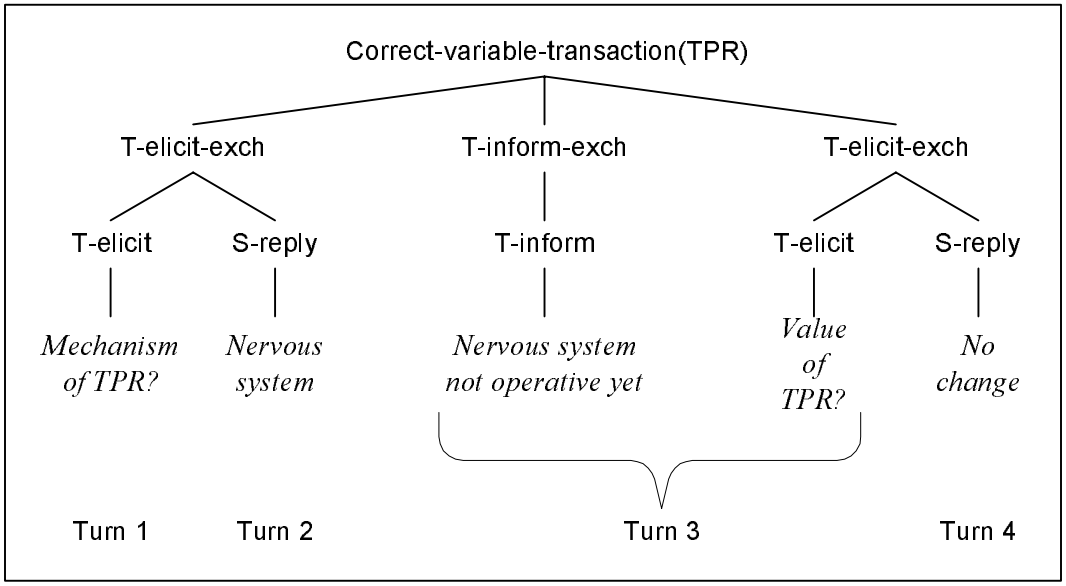

Figure 5.7: Initial discourse tree

This dialogue has been constructed from actual turns selected from different transcripts in order to show a large number of phenomena in a short text. In turn 2, the student gives a wrong answer. In turn 3, the tutor decides to explicitly reject the student's wrong answer before continuing. In turn 4, the student gives an answer which is worthwhile but not the desired answer; the statement is true but it doesn't respond directly to the point the tutor is trying to make. So the tutor asks a follow-up question in turn 5. In fact, students usually respond extremely well to such follow-up questions, but for the sake of the example we will assume that the student asks for help instead. The tutor could respond in a number of ways, such as with an explanation or another tutoring plan. In this case the tutor decides to give the student the answer and go on. In the continuation of turn 7, the tutor goes on to the remainder of the original tutoring plan.

Figure 5.8 shows the agenda of the tutoring planner during the planning of turn 3. An additional planning operator, *T-denies*, has been added as part of a revised plan for

T-teaches. The purpose of the additional operator is to create an explicit rejection of the previous wrong answer. Note that the *T-elicits* goal in Figure 5.8 corresponds to the question in turn 3. The question in turn 1 is no longer shown because it is no longer part of the current plan for *T-teaches(has-mechanism(TPR, neural))*. However, text has already been generated for this goal. This text is still part of the conversation and could possibly be involved in future decisions about topics such as pronoun usage. It is to handle this situation that we maintain two knowledge structures, one for the current tutorial goals and one for the evolving conversation.

Suppose instead of revising the plan for *T-teaches*, the tutor had decided to revise the parent goal instead, by choosing a new plan for *correct-variable*. For example, if the student is having trouble, the tutor might replace a plan involving questions with an easier plan involving mostly explanations. In that case, both of the younger siblings of *T-teaches* would be removed from the agenda without ever having been expanded. The replanning rules in Section 5.3.5 enable us to continue a coherent conversation regardless of the student's input.

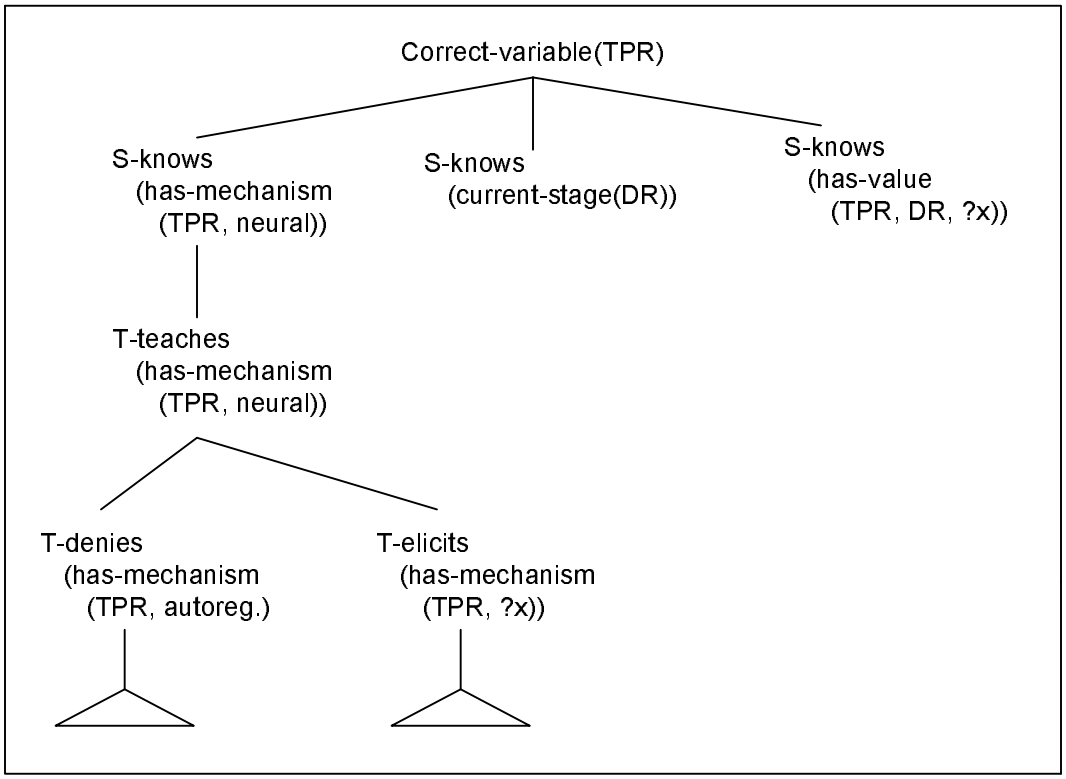

Figure 5.8: Tutoring agenda with rebuttal

Finally, Figure 5.9 shows the first seven turns of the completed conversation. (The others are omitted for reasons of space.) Turns 1 and 2 still belong to this transaction even though they were not part of the final plan for teaching the student about the value of TPR. In this diagram one can also see how the rejection of the student's error (*T-denies*) which was added during plan repair becomes an *T-inform* exchange, while the student's statement in turn 6 generates an *S-inform* exchange. We do not always know until after a response has been interpreted what the most logical place is to hook it into the discourse tree.

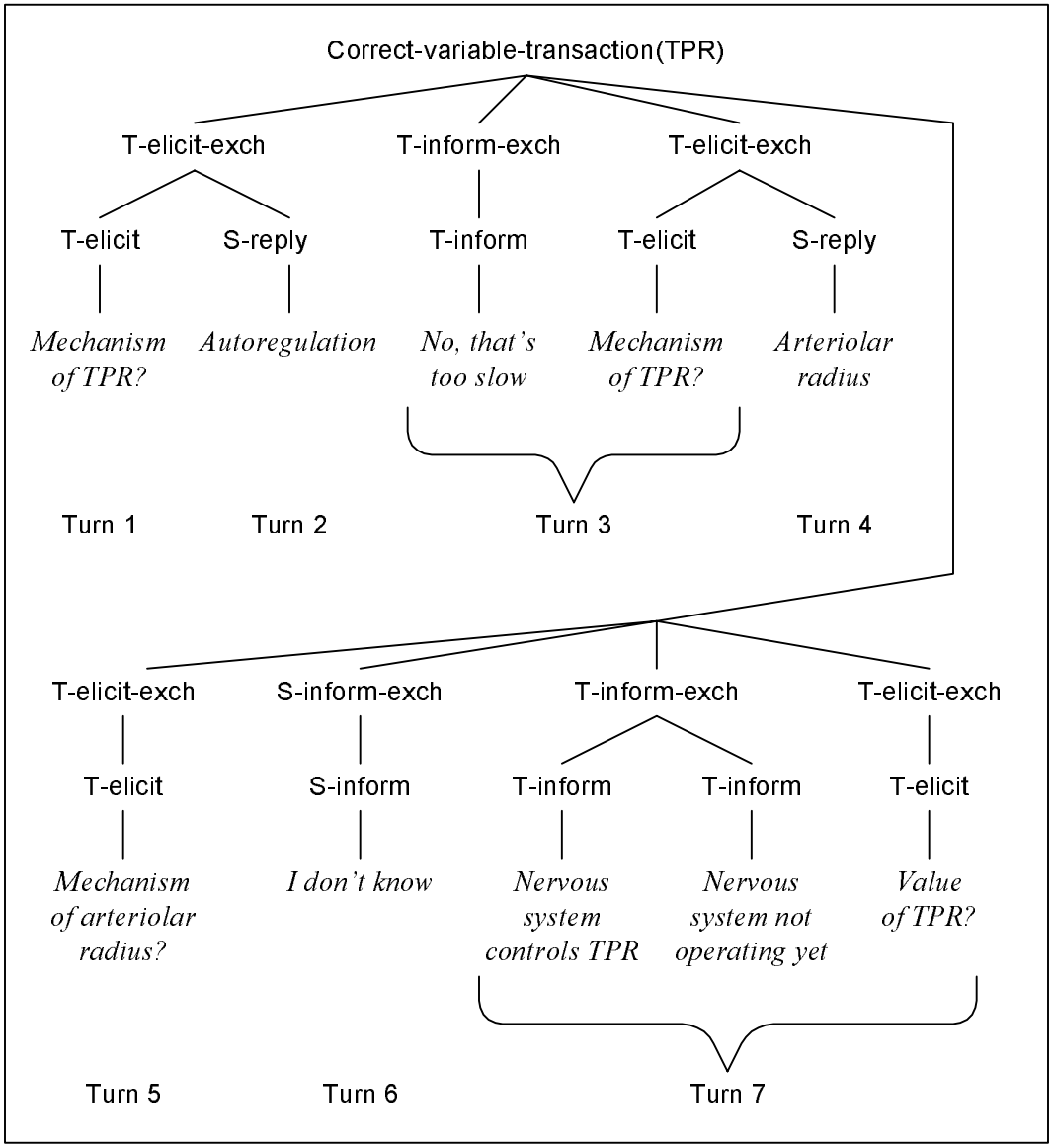

Figure 5.9: Final discourse tree

Chapter 6
Conclusions

The only reason for writing anything is to answer one's own questions. In the process of finding the answers, however incomplete or provisional, you might answer someone else's questions.

—Stanley Morison

In this section we present a summary of our results. We describe our conclusions about the structure of tutoring discourse and outline the architecture we have proposed for modeling these results in a large-scale, practical ITS. We summarize the main accomplishments of this research and propose short-term goals and long-term directions for future research.

6.1 Summary

We began this work by stating that our goal was to demonstrate how the use of theories and methods from the field of natural language generation could be used to improve the quality of text generated by an ITS.

We accomplished this goal by dividing it into three subgoals:

- Developing a model of the dialogue of human tutors based on current research in natural language generation.
- Analyzing a corpus of human-to-human tutoring sessions in cardiovascular physiology in terms of this model.
- Designing a large-scale, practical ITS which implements the model.

Using the corpus analysis and an analysis of the output of an older version of the CIRCSIM-Tutor system, we developed a list of requirements for a new ITS. We concluded that an architecture consisting of a high-level discourse planner and a turn planner running in parallel, rather than a separate pedagogical planner and a discourse planner, would best meet the requirements.

The following two sections summarize our major conclusions about the language used by the tutors in the transcripts. They can be loosely divided into two categories, depending on whether they relate to the hierarchical structure of the conversation or the interplay between tutor and student. These sections are followed by a brief summary of the proposed architecture.

6.1.1 Conclusions about the hierarchical structure of tutoring discourse

The following points summarize our major conclusions about the hierarchical aspects of the conversation:

- The tutor maintains global control of the conversation while responding turn by turn to the student's utterances.
- The global plan is shown by the hierarchical structure of the dialogue, which contains the following levels: physiological stage, core variable, attempts to teach each variable.
- Only the variables which the student missed are discussed, in a sequence related to a solution trace for the problem.
- The tutor has a variety of methods for teaching the value of each variable, including both single-turn and multi-turn plans.
- Methods may be nested. If the student gives an incorrect answer, the tutor has several options:
    - Give the student the answer and go on
    - Provide the student with additional information and try again
    - Try a new method at any level

- Most variables are corrected using one of five schemata. These schemata are used for the following cases:
    - Primary variable (i.e. first variable in the solution trace)
    - First incorrect neural variable
    - Subsequent incorrect neural variable
    - Non-neural variable, taught based on its determinants
    - Non-neural variable, taught based on the value of an earlier variable in the conversation

6.1.2 Conclusions about dialogue-related aspects of tutoring discourse

The following points summarize our major conclusions about the dialogue-related aspects of the conversation:

- Each turn has the following structure. Each of the sections is optional.

    Response to student's previous statement
        Acknowledgment of student's statement
            (e.g. *yes*, *no*, etc.)
        Content-oriented reply
    New material

- A turn ends with a question for the student (or an imperative). Usually the question or imperative is explicit, but sometimes the tutor makes a statement from which the student can infer the type of response desired.[1]

- Acknowledgments are interpersonal transactions which are domain-independent. When the tutor needs to convey domain knowledge as part of the reply, the content-oriented reply section is used. The type of reply depends on the type of the student's utterance. Some of the more common types of student utterances include:
    - Correct answer
    - Wrong answer
    - Hedged answer, right or wrong
    - Linguistically close but not exact answer

[1] CIRCSIM-Tutor will not emulate these indirect requests because students expect explicit requests from a program. See Section 5.3.7.

- A step toward the correct answer
- Student initiative (i.e. student changes the topic)
- A combination of the above

Some of the more common types of content-oriented replies include:

- A restatement of the student's statement in more precise language
- A statement supporting or denying the student's statement
- An interactive formula which points out a contradiction in the student's utterances

- After replying to the student's statement, the tutor may continue with the next step from the teaching plan. If no new material is included, then the content-oriented reply must be present and must end with a question.

6.1.3 Summary of proposed architecture for a text-based ITS

This section lists the salient points of the architecture proposed in Chapter 5.

- The system is divided into two routines running in parallel, the *tutorial planner*, which makes discourse decisions for units larger than a turn, and the *turn planner*, which assembles individual turns. The tutorial planner generates a series of semantic forms. The turn planner collects the semantic forms for a turn and generates text for them as a unit. The turn planner is also responsible for connecting the turn to the ongoing conversation.

- It can maintain a dialogue, including appropriate responses to wrong or partially correct responses on the part of the student, while carrying out a global tutoring plan. It can use multi-turn plans and amend or drop them if the student does not respond as expected.

- It can provide specific content-based responses to common student errors.

- It can use the same input to generate multiple tutorial discourse phenomena similar to the hints, explanations and interactive sequences generated by expert human tutors.

- It can teach the same concept in multiple ways.

- It can say the same thing in multiple ways.

6.2 Contributions of this work

The following list summarizes the major accomplishments of this dissertation:

- Establishing a new model for conversational turns in CIRCSIM-Tutor through the use of Conversation Analysis.
    - Identification of the acknowledgment, the content-oriented reply and new material from the tutorial plan as potential parts of every turn.
    - Distinguishing the functions of the acknowledgment and the content-oriented reply.
    - Initial characterization of the types of content-oriented reply.

- Establishing a knowledge representation for tutorial knowledge, namely a schema which is a sequence of discourse goals.
    - Demonstration that tutorial knowledge ("how to tutor") cannot be derived from domain knowledge ("what to tutor").
    - Identification of the basic schemata required for correcting variables, including a schema, tutoring non-neural variables based on variables tutored earlier in the conversation, which had not been previously described.
    - Description of new schemata for replying to student utterances, such as *show-contradiction* and the pseudo-diagnostic question.

- Characterizing crucial aspects of system design for an interactive natural-language ITS.
    - Demonstration that text generation provides a superior model for a text-based ITS, i.e. that the fundamental distinction in CIRCSIM-Tutor should be between the tutorial planner and the turn planner, not between the pedagogical planner and the discourse planner.
    - Demonstration that text must be generated a turn at a time in order to ensure coherent turns.
    - Characterization of the dynamic behavior of the tutorial planner: how to integrate responses to the student with the global plan, how to backtrack when the student makes a mistake, how to use multi-turn plans and how to abandon them when necessary, how to respond to student initiatives.
    - Characterization of the behavior of the turn planner, including a description of how the turn planner builds and uses a discourse tree separate from the plan history in order to ensure inter-turn coherence.

- Characterizing the semantic primitives required to implement the system.
    - Identification a list of potential semantic primitives abstracted from the transcripts.
    - Identification of a list of basic semantic primitives sufficient for a prototype.
    - Demonstration of how to derive multiple observed phenomena, such as hints, explanations and interactive explanations, as different ways of realizing the same schema.

- Contributing to a deeper understanding of phenomena in the CIRCSIM-Tutor transcripts previously described only at the surface level.
    - Definition of previously described phenomena, i.e. acknowledgments, hints, explanations, interactive explanations, summaries and student initiatives, in a manner suitable for text generation
    - Demonstration that identifying which part of a turn an observed phenomena belongs to, i.e. response to the student or new material, is essential to the description.

- Contributing to the understanding of the potential capabilities and limitations of v. 3 of CIRCSIM-Tutor.
    - Identification of key factors, i.e. good language, variety, interactivity, and coherence, which influence student understanding and retention.
    - Identification of methods for using deeper levels of the domain concept map in dialogue.
    - Identification of places where a functional model of the domain could contribute to CIRCSIM-Tutor.

- Developing a declarative model of the domain knowledge required for CIRCSIM-Tutor

- Contributing to the understanding of tradeoffs in ITS design
    - Cooperative conversation vs. mainly tutor-led conversation
    - Full vs. restricted student initiative processing
    - Free-text input vs. short-answer questions
    - Use of turn-taking rules vs. use of explicit questions
    - Plan recognition vs. goal recognition
    - Depth vs. breadth of generation

6.3 Feasible enhancements using the current architecture

6.3.1 Creating realistic tutorial and linguistic knowledge bases

In order to model the syntactic and lexical usage of the expert tutors as well as their high-level pedagogical and discourse goals, we need to mine lexical and syntactic information from the transcripts. Although this process is tedious, there are several advantages to doing it this way instead of by introspection. In addition to greater fidelity to what human tutors actually do, we obtain more natural speech patterns. We also obtain a greater variety of constructions, since introspection is difficult in a realm such as lexical choice where humans usually operate without much conscious thought.

In addition to identifying concepts used by the tutors and the syntactic forms used to express them, it would be worthwhile to identify ways in which the tutors combine semantic forms into sentences. This information would enable us to expand the list of combination rules in the turn planner and thus generate more complex turn structures.

6.3.2 Using CIRCSIM-Tutor to study discourse rules

Many types of linguistic rules, such as those governing change of topic and focus, the introduction of discourse particles, and the combination of concepts into sentences, have not yet been formulated in a precise enough form for text generation. CIRCSIM-Tutor provides an opportunity to write and test such rules. In particular, a worthwhile topic to study is the degree to which such rules can be deduced by the turn planner instead of being specified in a tutorial planning schema.

6.3.3 Implementing opportunistic planning

The use of opportunistic planning provides an opportunity to handle some categories of student initiatives. There are two places where opportunistic planning could

be used to shorten the conversation if desired. First, when a student gives the correct value for one variable while the tutor is trying to elicit the value for another variable, we could accept this information instead of ignoring it. Second, when a student uses a phrase like "Now I get it" to indicate that the rest of the argument is now superfluous, the tutor could skip the rest of the current variable or another appropriate subsection. However, our pedagogical consultants are currently opposed to the use of tutoring methods which do not require an explicit conversation for each variable.

6.3.4 Adding functional knowledge to the domain knowledge base

The addition of a functional model to the domain knowledge base would allow the generation of more complex explanations. With respect to lexicon and syntax, it is generally a good idea not to generate text which we could not understand if the student chooses to use it. However, this principle does not hold for larger concepts because the student's response will always be guided by a question which we have asked. Thus it is feasible for us to generate text based a functional model even though we would not be able to understand such a functional explanation as input. For example, a physical metaphor for some aspect of blood flow might be useful to the student as part of a generated explanation even if we are not prepared to discuss it.

6.3.5 Improving error handling through increased use of intention-based planning

In the current design, in order for CIRCSIM-Tutor to give a specific response to a student error, it needs an adjunct schema tailored to that error. Although schemata are a good representation for CIRCSIM-Tutor's methods of teaching new material, greater use of intention-based planning during error handling would permit us to provide specific responses to a larger number of errors.

It is not yet feasible to implement deep intention-based planning for CIRCSIM-Tutor as a whole. At the tutorial level, even expert teachers cannot often explain why they teach the way they teach. At the turn planning level, the axioms relating most types of phrases and clauses to communicative goals have not yet been worked out. Although some small-scale systems have been created, the technology is not yet feasible for a broad-coverage system.

6.4 Evaluating potential long-term research directions

6.4.1 Free-text input, student initiatives and plan recognition

CIRCSIM-Tutor is primarily a tutor-driven system although it can handle simple student initiatives such as requests for help. One reason we do not usually ask open-ended questions in order to diagnose a student's misconceptions is that we cannot understand totally free text, especially when the student's statement does not make sense according to our domain model.[2] We cannot handle free-text student initiatives for the same reason. Finally, we cannot handle interruptions which need to start a stacked discourse plan. These restrictions, which are acceptable in our intended application, express the limitations of the CIRCSIM-Tutor model. Putting these facts together, CIRCSIM-Tutor can handle most things which happen in a tutor-led tutoring session, but it cannot handle a true cooperative conversation with two independently planning agents.

SCHOLAR [Carbonell 1970] could switch back and forth between tutor-led and student-led sections of a conversation. We can do "mixed-initiative" processing in this sense. Allowing a fully general form of student initiative, i.e. letting the student change the

[2] A second reason is that many of the misconceptions involve functional roles, which our domain model does not support. Since we have many other sources of information (see Section 5.3.4), we do not need an explicit diagnosis to respond appropriately to the student.

topic, would require the ability to handle a fully cooperative conversation and thus the need to maintain multiple discourse contexts.

Since the ability to understand unrestricted free-text input is beyond the capability of a working ITS and the technology for using multi-agent plans is still experimental, we conclude that the ability to handle a fully cooperative conversation is too difficult for a broad-coverage ITS. As the tutor knows what needs to be corrected, it is not unreasonable for the tutor to be in charge of the conversation. Additionally, empirical data collected from live tutors shows that they maintain a general control over the conversation even when they give the student a certain amount of leeway.[3]

The importance of cooperative conversation is not in the conversation itself, but in interactivity, which keeps the student actively involved with the material. Therefore we have suggested several ways to use the text generation capability of the ITS to increase interactivity in spite of the unavoidable limitations on natural language understanding. CIRCSIM-Tutor has several ways of using language which increase interaction with the student but don't require full interactivity. First, through the use of schemata, we can achieve both pedagogical and linguistic variety, i.e. we have a variety of ways to teach a concept and a variety of ways to express each of them. Second, we have a variety of ways to respond to student errors and other types of unexpected input.

Plan recognition involves inferring the other speaker's plans from his or her utterances [Carberry 1990]. Alternative approaches have been suggested by Traum and Hinkelman [1992] and McRoy [McRoy & Hirst 1995]. The former especially is less computationally intensive. As long as conversations are led by the tutor and every turn must end in an explicit question or request, plan recognition is not a major issue. The

[3] Michael Glass points out that the best-known AI program which allows the student to lead the conversation is ELIZA [Weizenbaum 1967]. Unfortunately ELIZA is of limited value as a tutor.

student is expected to follow the tutor's lead, not to carry out an independent discourse plan. When the student does not respond to the immediately preceding question, we can generally identify the student's goal from a short list which includes responding to a higher-level goal on the agenda, requesting a definition, asking for help, or hedging an otherwise direct response.

In summary, modeling the student's goals as well as the tutor's would require a significantly more complex architecture. While this is an interesting and worthwhile research objective, it is probably too complex to contribute much to a working ITS in the near future.

6.4.2 Application to other genres and other languages

Nothing in the CIRCSIM-Tutor design is specific to cardiovascular physiology. With suitable knowledge base entries for the rhetorical forms needed, CIRCSIM-Tutor could be used to generate text in other genres. Additionally, again due to the modular nature of the design, generating text in another language would require only a new grammar and lexicon.

6.5 Conclusions

The purpose of this dissertation was to demonstrate the utility of natural language generation as the underlying model for an intelligent tutoring system (ITS) in cardiovascular physiology. To achieve that goal, we divided it into three subgoals: developing a model of the tutorial dialogue of human tutors, analyzing a corpus of human-to-human tutoring sessions in cardiovascular physiology in terms of this model, and designing an ITS which implements the model. We have given a detailed analysis of our corpus using this model, including a discussion of how tutors sequence their corrections, begin and end phases of the discourse, acknowledge responses, convey information to the

student, provide hints, conduct interactive explanations and switch between domain models. We have presented a design for an ITS which uses this model to show that it can be implemented with current technology.

We expect this model to produce longer, more complex and more varied dialogues than previous work. Version 3 can handle nested goals, recovery from student errors, simple student initiatives, and many other phenomena which its predecessor could not. Additionally, we hope that people will find CIRCSIM-Tutor useful as a testbed for theories about discourse, including areas such as topic/focus, given/new, and the use of discourse markers, especially in situations involving dialogues. Finally, we hope that CIRCSIM-Tutor will be useful to the medical students who inspired it.

References

Appelt, Douglas E. 1985. *Planning English Sentences*. Cambridge: Cambridge University Press. An earlier version was published as the author's Ph.D. thesis at Stanford University in 1981.

Austin, J. L. 1962. *How to Do Things with Words*. Oxford: Clarendon Press.

Barrett, Anthony, Keith Golden, Scott Penberthy and Daniel Weld. 1994. *UCPOP User's Manual*. Technical Report 93–09–06, Department of Computer Science and Engineering, University of Washington.

Bilange, Eric. 1991. "A Task Independent Oral Dialogue Model," *Proceedings of the Fifth Conference of the European Chapter of the Association for Computational Linguistics*, Berlin, pp. 83–88.

Borchardt, Gary C. 1994. *Thinking Between the Lines: Computers and the Comprehension of Causal Descriptions*. Cambridge, MA: MIT Press.

Brown, John S., Richard R. Burton and Frank Zdybel. 1973. "A Model-Driven Question-Answering System for Mixed-Initiative Computer-Assisted Instruction," *IEEE Transactions on Systems, Man, and Cybernetics* 3(3): 248–257.

Carberry, Sandra. 1990. *Plan Recognition in Natural Language Dialogue*. Cambridge, MA: MIT Press.

Carbonell, Jaime R. 1970. "AI in CAI: Artificial Intelligence Approach to Computer Assisted Instruction," *IEEE Transactions on Man-Machine Systems* 11(4): 190–202.

Cawsey, Alison. 1992. *Explanation and Interaction: The Computer Generation of Explanatory Dialogues*. Cambridge, MA: MIT Press.

Chevallier, R. 1992. "Studia: un système tutoriel coopératif fondé sur la négotiation et sur un modèle dynamique de dialogue." In [Frasson, Gauthier & McCalla 1992, pp. 58–65].

Chu-Carroll, Jennifer and Sandra Carberry. 1995. "Response Generation in Collaborative Negotiation," *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, Cambridge, pp. 136–143.

Clancey, William J. 1987. *Knowledge-Based Tutoring: The GUIDON Program*. Cambridge, MA: MIT Press. An earlier version was published as the author's 1979 Ph.D. thesis at Stanford University.

Cohen, Philip R. and C. Raymond Perrault. 1979. "Elements of a Plan-Based Theory of Speech Acts," *Cognitive Science* 3(3): 177–212. Reprinted in [Grosz, Sparck Jones & Webber 1986]. Based on Cohen's 1978 Ph.D. thesis at the University of Toronto.

Cohen, Philip R., C. Raymond Perrault and James F. Allen. 1982. "Beyond Question Answering." In Wendy G. Lehnert and Martin H. Ringle, eds., *Strategies for Natural Language Processing*, Hillsdale, NJ: Lawrence Erlbaum, pp. 245–274.

Collins, Allan, Eleanor H. Warnock, Nelleke Aiello and Mark L. Miller. 1975. "Reasoning from Incomplete Knowledge." In Daniel G. Bobrow and Allan Collins, eds., *Representation and Understanding: Studies in Cognitive Science*, New York: Academic Press, pp. 383–415.

Collins, Allan, Eleanor H. Warnock and Joseph J. Passafiume. 1975. "Analysis and Synthesis of Tutorial Dialogues." In Gordon H. Bower, ed., *The Psychology of Learning and Motivation*, v. 9, New York: Academic Press, pp. 49–87.

Collins, Allan. 1977. "Processes in Acquiring Knowledge." In Richard C. Anderson, Rand J. Spiro and William E. Montague, eds., *Schooling and the Acquisition of Knowledge*. Hillsdale, NJ: Lawrence Erlbaum, pp. 339–363.

Collins, Allan and Albert L. Stevens. 1982. "Goals and Strategies of Inquiry Teachers." In Robert Glaser, ed., *Advances in Instructional Psychology, v. 2*, Hillsdale, NJ: Lawrence Erlbaum, pp. 65–119.

Collins, Allan and Albert L. Stevens. 1991. "A Cognitive Theory of Inquiry Teaching." In Peter Goodyear, ed., *Teaching Knowledge and Intelligent Tutoring*, Norwood, NJ: Ablex, pp. 203–230. Earlier version published in Charles M. Reigeluth, ed., *Instructional Design: Theories and Methods*.

Dahlbäck, Nils and Arne Jönsson. 1989. "Empirical Studies of Discourse Representations for Natural Language Interfaces," *Proceedings of the Fourth Conference of the European Chapter of the Association for Computational Linguistics*, Manchester, pp. 291–298.

Dale, Robert, Eduard Hovy, Dietmar Rösner and Oliviero Stock, eds. 1992. *Aspects of Automated Natural Language Generation*. Proceedings of the Sixth International Workshop on Natural Language Generation, Trento, Italy. (Springer-Verlag Lecture Notes in AI, no. 587.) Berlin: Springer-Verlag.

Danlos, Laurence. 1987. *The Linguistic Basis of Text Generation*. Cambridge: Cambridge University Press. Originally published in French in 1985.

Dickinson, C. J., C. H. Goldsmith and David L. Sackett. 1973. "MACMAN: A Digital Computer Model for Teaching Some Basic Principles of Hemodynamics," *Journal of Clinical Computing* 2(4): 42–50.

Elhadad, Michael. 1992. *Using Argumentation to Control Lexical Choice: A Functional Unification Implementation*. Ph.D. thesis, Department of Computer Science, Columbia University.

Elhadad, Michael. 1993. *FUF: The Universal Unifier, User Manual, Version 5.2.* Available from Department of Computer Science, Ben Gurion University of the Negev.

Evens, Martha W., John Spitkovsky, Patrick Boyle, Joel A. Michael and Allen A. Rovick. 1993. "Synthesizing Tutorial Dialogues," *Proceedings of the 15th Annual Conference of the Cognitive Science Society*, Boulder. Hillsdale, NJ: Lawrence Erlbaum.

Fawcett, Robin P. 1987. "Semantics of Clause and Verb for Relational Processes in English." In M. A. K. Halliday and Robin P. Fawcett, eds., *New Developments in Systemic Linguistics, v. 1: Theory and Description*, London: Pinter, pp. 130–183.

Fikes, Richard E. and Nils J. Nilsson. 1971. "STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving," *Artificial Intelligence* 2(3–4): 189–208.

Fox, Barbara. 1987. *Discourse Structure and Anaphora: Written and Conversational English*. Cambridge: Cambridge University Press.

Fox, Barbara. 1993. *The Human Tutorial Dialogue Project*. Hillsdale, NJ: Lawrence Erlbaum.

Frasson, Claude, Gilles Gauthier and Gordon I. McCalla, eds. 1992. *Intelligent Tutoring Systems: Second International Conference (ITS '92), Proceedings*, Montreal. (Springer-Verlag Lecture Notes in Computer Science, no. 608.) Berlin: Springer-Verlag.

Frasson, Claude, Gilles Gauthier and Alan Lesgold, eds. 1996. *Intelligent Tutoring Systems: Third International Conference (ITS '96), Proceedings*, Montreal. (Springer-Verlag Lecture Notes in Computer Science, no. 1086.) Berlin: Springer-Verlag.

Freedman, Reva. 1995. "Using Pedagogical Knowledge to Structure Text Generation in an Intelligent Tutoring System," *Proceedings of the 1995 Midwest Artificial Intelligence and Cognitive Science Society Conference*, Carbondale.

Freedman, Reva. 1996a. "Using a Text Planner to Model the Behavior of Human Tutors in an ITS." In Michael Gasser, ed., *Online Proceedings of the 1996 Midwest Artificial Intelligence and Cognitive Science Conference*, Bloomington, IN. URL http://www.cs.indiana.edu/event/maics96/Proceedings/Freedman/freedman.html *or* freedman.ps.

Freedman, Reva. 1996b. "Using Tutoring Patterns to Generate More Cohesive Text in an Intelligent Tutoring System" In Daniel C. Edelson and Eric A. Domeshek, eds., *International Conference on the Learning Sciences,* Proceedings of ICLS '96, Evanston, pp. 75–82.

Freedman, Reva and Martha W. Evens. 1996. "Generating and Revising Hierarchical Multi-turn Text Plans in an ITS." In [Frasson, Gauthier & Lesgold 1996, pp. 632–640].

Gerlach, Michael and Helmut Horacek. 1989. "Dialog Control in a Natural Language System," *Fourth Conference of the European Chapter of the Association for Computational Linguistics: Proceedings of the Conference*, Manchester, pp. 27–34.

Grice, H. Paul. 1975. "Logic and Conversation." In Peter Cole and Jerry L. Morgan, eds., *Speech Acts* (*Syntax and Semantics*, v. 3), New York: Academic Press, pp. 41–58.

Grosz, Barbara J. and Candace L. Sidner. 1986. "Attention, Intentions and the Structure of Discourse," *Computational Linguistics* 12(3): 175–204.

Grosz, Barbara J., Karen Sparck Jones and Bonnie Lynn Webber, eds. 1986. *Readings in Natural Language Processing*. Los Altos, CA: Morgan Kaufmann.

Halliday, M. A. K. and Ruqaiya Hasan. 1976. *Cohesion in English*. London: Longman.

Halliday, M. A. K. 1985. *An Introduction to Functional Grammar*. London: E. Arnold.

Hollan, James D., Edwin L. Hutchins and Louis Weitzman. 1984. "STEAMER: An Interactive Inspectable Simulation-Based Training System," *AI Magazine* 5(2): 15–27.

Helmut Horacek. 1992. "An Integrated View of Text Planning." In [Dale et al. 1992, pp. 29–44].

Hovy, Eduard H. 1988. "Planning Coherent Multisentential Text," *Proceedings of the 26th Annual Meeting of the Association for Computational Linguistics*, Buffalo.

Hovy, Eduard H. and Kathleen F. McCoy. 1989. "Focusing Your RST: A Step toward Generating Coherent Multisentential Text," *Program of the 11th Annual Conference of the Cognitive Science Society*, Montreal. Hillsdale, NJ: Lawrence Erlbaum.

Hovy, Eduard H. 1991. "Approaches to the Planning of Coherent Text." In [Paris, Swartout & Mann 1991, pp. 83–102].

Hovy, Eduard H., et al. 1992. "Employing Knowledge Resources in a New Text Planner Architecture." In [Dale et al. 1992, pp. 57–72].

Huang, Xueming and Gordon I. McCalla. 1992. "Instructional Planning using Focus of Attention." In [Frasson, Gauthier & McCalla 1992, pp. 443–450].

Hume, Gregory D. 1995. *Using Student Modelling to Determine When and How to Hint in an Intelligent Tutoring System*. Ph.D. thesis, Department of Computer Science, Illinois Institute of Technology.

Hume, Gregory D., Joel A. Michael, Allen A. Rovick and Martha W. Evens. 1993. "The Use of Hints as a Tutorial Tactic," *Proceedings of the 15th Annual Conference of the Cognitive Science Society*, Boulder. Hillsdale, NJ: Lawrence Erlbaum.

Hume, Gregory D., Joel A. Michael, Allen A. Rovick and Martha W. Evens. 1995. "Controlling Active Learning: How Tutors Decide When to Generate Hints," *Proceedings of the 8th Florida Artificial Intelligence Research Symposium*, Pensacola.

Hume, Gregory D., Joel A. Michael, Allen A. Rovick and Martha W. Evens. 1996. "Hinting as a Tactic in One-on-One Tutoring," *Journal of the Learning Sciences* 5(1): 32–47.

Jönsson, Arne. 1991. "A Dialogue Manager using Initiative-Response Units and Distributed Control," *Proceedings of the Fourth Conference of the European Chapter of the Association for Computational Linguistics*, Manchester, pp. 233-238.

Jullien, Cléo and Jean-Charles Marty. 1989. "Plan Revision in Person-Machine Dialog," *Fourth Conference of the European Chapter of the Association for Computational Linguistics: Proceedings of the Conference*, Manchester, pp. 153–160.

Kempen, Gerard. 1987. *Natural Language Generation: New Results in AI, Psychology and Linguistics*. (NATO ASI Series, series E: Applied Science, no. 136.) Dordrecht: Martinus Nijhoff.

Khuwaja, Ramzan A. 1994. *A Model of Tutoring: Facilitating Knowledge Integration using Multiple Models of the Domain*. Ph.D. thesis, Department of Computer Science, Illinois Institute of Technology.

Khuwaja, Ramzan A., Allen A. Rovick, Joel A. Michael and Martha W. Evens. 1995. "A Tale of Three Protocols: The Implications for Intelligent Tutoring Systems." In E. A. Yfantis, ed., *Intelligent Systems: Third Golden West International Conference*, Las Vegas, 1994 (Theory and Decision Library, Series D: System Theory, Knowledge Engineering, and Problem Solving, v. 15), Dordrecht: Kluwer Academic.

Kim, Nakhoon. 1989. *CIRCSIM-Tutor: An Intelligent Tutoring System for Circulatory Physiology*. Ph.D. thesis, Department of Computer Science, Illinois Institute of Technology.

Levinson, Stephen C. 1983. *Pragmatics*. Cambridge: Cambridge University Press.

Li, Jun, Jai Seu, Martha W. Evens, Joel A. Michael and Allen A. Rovick. 1992. "Computer Dialogue System (*CDS*): A System for Capturing Computer-Mediated Dialogue", *Behavior Research Methods, Instruments, & Computers* 24(4): 535–540.

McCoy, Kathleen F. and Jeanette Cheng. 1991. "Focus of Attention: Constraining What Can Be Said Next." In [Paris, Swartout & Mann 1991, pp. 103–124].

McDermott, Drew. 1981. "Artificial Intelligence Meets Natural Stupidity." In John Haugeland, ed., *Mind Design: Philosophy, Psychology, Artificial Intelligence*, Cambridge, MA: MIT Press, pp. 143–160.

McKeown, Kathleen R. 1985. *Text Generation: Using Discourse Strategies and Focus Constraints to Generate Natural Language Text*. Cambridge: Cambridge University Press. An earlier version was published as the author's Ph.D. thesis at the University of Pennsylvania in 1982.

McRoy, Susan W. and Graeme Hirst. 1995. "The Repair of Speech Act Misunderstandings by Abductive Inference," *Computational Linguistics* 21(4): 435–478.

Maier, Elisabeth A. and Eduard H. Hovy. 1991. "A Metafunctionally Motivated Taxonomy for Discourse Structure Relations," *Proceedings of the Third European Workshop on Text Generation*, Innsbruck.

Mann, William C. 1985. "An Introduction to the Nigel Text Generation Grammar." In James D. Benson and William S. Greaves, eds., *Systemic Perspectives on Discourse, v. 1: Selected Theoretical Papers from the Ninth International Systemic Workshop*, Toronto, 1982 (*Advances in Discourse Processes*, v. 15), Norwood, NJ: Ablex, pp. 84–95.

Mann, William C. and James A. Moore. 1981. "Computer Generation of Multiparagraph English Text," *American Journal of Computational Linguistics* 7(1): 17–29.

Mann, William C. and Sandra A. Thompson. 1986. "Relational Propositions in Discourse," *Discourse Processes* 9: 57–90.

Mann, William C. and Sandra A. Thompson. 1988. "Rhetorical Structure Theory: Toward a Functional Theory of Text Organization," *Text* 8(3): 243–281.

Martin, Jim R. 1983. "Conjunction: The Logic of English Text." In János S. Petöfi and Emil Sözer, eds., *Micro and Macro Connexity of Texts* (*Papiere zur Textlinguistik*, v. 45), Hamburg: Helmut Burke Verlag, pp. 1–72.

Maybury, Mark T. 1992. "Communicative Acts for Explanation Generation," *International Journal of Man-Machine Studies* 37(2): 135–172.

Maybury, Mark T. 1991. *Planning Multisentential English Text Using Communicative Acts*. Ph.D. thesis, University of Cambridge Computer Laboratory. Technical Report 239.

Meteer, Marie W. 1992. *Expressibility and the Problem of Efficient Text Planning*. London: Pinter. An earlier version was published as the author's Ph.D. thesis at the University of Massachusetts at Amherst in 1990.

Moore, Johanna D. 1995. *Participating in Explanatory Dialogues: Interpreting and Responding to Questions in Context*. Cambridge, MA: MIT Press. An earlier version was published as the author's 1989 Ph.D. thesis at UCLA.

Moore, Johanna D. and Cécile L. Paris. 1989. "Planning Text for Advisory Dialogues," *Proceedings of the 27th Annual Meeting of the Association for Computational Linguistics*, Vancouver.

Moore, Johanna D. and Martha Pollack. 1993. "A Problem for RST: The Need for Multi-Level Discourse Analysis," *Computational Linguistics* 18(4): 537–544.

Moore, Johanna D. and William R. Swartout. 1991. "A Reactive Approach to Explanation: Taking the User's Feedback into Account." In [Paris, Swartout & Mann 1991, pp. 3–48].

Neches, Robert, William R. Swartout and Johanna D. Moore. 1985. "Enhanced Maintenance and Explanation of Expert Systems through Explicit Models of their Development," *IEEE Transactions on Software Engineering*, SE-11(11): 1337–1351.

Paris, Cécile L. 1985. "Description Strategies for Naive and Expert Users," *Proceedings of the 23rd Annual Meeting of the Association for Computational Linguistics*, Chicago.

Paris, Cécile L. 1988. "Tailoring Object Descriptions to a User's Level of Expertise," *Computational Linguistics* 14(3): 64–78.

Paris, Cécile L. 1991. "Generation and Explanation: Building an Explanation Facility for the Explainable Expert Systems Framework." In [Paris, Swartout & Mann 1991, pp. 49–82].

Paris, Cécile L. 1993. *User Modelling in Text Generation*. London: Pinter. An earlier version was published as the author's Ph.D. thesis at Columbia University in 1987.

Paris, Cécile L. and Kathleen R. McKeown. 1987. "Discourse Strategies for Describing Complex Physical Objects." In [Kempen 1987, pp. 97–115].

Paris, Cécile L., William R. Swartout and William C. Mann, eds. 1991. *Natural Language Generation in Artificial Intelligence and Computational Linguistics*. Proceedings of the Fourth International Workshop on Text Generation, Catalina Island, CA, 1988. Boston: Kluwer.

Darwyn R. Peachey and Gordon I. McCalla. 1986. "Using Planning Techniques in Intelligent Tutoring Systems," *International Journal of Man-Machine Studies*, 24(1): 77–98.

Penberthy, J. Scott and Daniel S. Weld. 1992. "UCPOP: A Sound, Complete, Partial-Order Planner for ADL," *Proceedings of the Third International Conference on Knowledge Representation and Reasoning (KR-92)*, Cambridge, MA.

Reiter, Ehud. 1991. "A New Model of Lexical Choice for Nouns," *Computational Intelligence* 7(4): 240–251.

Reiter, Ehud. 1994. "Has a Consensus NL Generation Architecture Appeared, and is it Psycholinguistically Plausible?," *Proceedings of the Seventh International Workshop on Natural Language Generation*, Kennebunkport, ME, pp. 163–170. Also available from the `cmp-lg` archive as paper 9411032.

Reiter, Ehud. 1995. "NLG vs. Templates," *Proceedings of the Fifth European Workshop on Natural Language Generation*, Leiden. Also available from the `cmp-lg` archive as paper 9504013.

Reiter, Ehud. 1996. "Building Natural-Language Generation Systems." In Alison Cawsey, ed., *Proceedings of the AI and Patient Education Workshop*, Glasgow, GIST Technical Report G95.3, Department of Computing Science, University of Glasgow. Also available from the `cmp-lg` archive as paper 9605002.

Resnick, Lauren B. 1977. "Holding an Instructional Conversation: Comments on Chapter 10 by Collins." In Richard C. Anderson, Rand J. Spiro and William E. Montague, eds., *Schooling and the Acquisition of Knowledge*. Hillsdale, NJ: Lawrence Erlbaum, pp. 365–372.

Robin, Jacques. 1994. *Revision-Based Generation of Natural Language Summaries Providing Historical Background: Corpus-Based Analysis, Design, Implementation and Evaluation*. Ph.D. thesis, Department of Computer Science, Columbia University. Technical Report CUCS-034-94.

Rösner, Dietmar and Manfred Stede. 1992. "Customizing RST for the Automatic Production of Technical Manuals." In [Dale et al. 1992, pp. 199–214].

Rovick, Allen A. and Lisa Brenner. 1983. "HEARTSIM: A Cardiovascular Simulation with Didactic Feedback," *The Physiologist* 26(4): 236–239.

Rovick, Allen A. and Joel A. Michael. 1992. "The Predictions Table: A Tool for Assessing Students' Knowledge," *American Journal of Physiology*, 263(6, part 3): S33–S36. Also available as *Advances in Physiology Education*, 8(1): S33–S36.

Sacerdoti, Earl D. 1974. "Planning in a Hierarchy of Abstraction Spaces," *Artificial Intelligence* 5(2): 115–135.

Sacerdoti, Earl D. 1977. *A Structure for Plans and Behavior*. New York: Elsevier.

Sacks, Harvey, Emanuel A. Schegloff and Gail Jefferson. 1974. "A Simplest Systematics for the Organization of Turn-Taking in Conversation," *Language* 50(4): 696–735.

Sanders, Gregory A. 1995. *Generation of Explanations and Multi-Turn Discourse Structures in Tutorial Dialogue, Based on Transcript Analysis*. Ph.D. thesis, Department of Computer Science, Illinois Institute of Technology.

Schegloff, Emanuel A. and Harvey Sacks. 1973. "Opening up Closings," Semiotica 7(4): 289–327.

Schiffrin, Deborah. 1987. *Discourse Markers*. Cambridge: Cambridge University Press.

Searle, John R. *Speech Acts*. 1969. Cambridge: Cambridge University Press.

Seu, Jai, Ru-Charn Chang, Jun Li, Martha W. Evens, Joel A. Michael and Allen A. Rovick. 1991. "Language Differences in Face-to-Face and Keyboard-to-Keyboard Tutoring Sessions," *Proceedings of the 13th Annual Conference of the Cognitive Science Society*, Chicago. Hillsdale, NJ: Lawrence Erlbaum.

Sidner, Candace L. 1983. "Focusing in the Comprehension of Definite Anaphora." In Michael Brady and Robert C. Berwick, eds., *Computational Models of Discourse*, Cambridge, MA: MIT Press, pp. 267–330. Reprinted in [Grosz, Sparck Jones & Webber 1986]. Based on the author's 1979 Ph.D. thesis at MIT.

Sinclair, John M. and Richard M. Coulthard. 1975. *Towards an Analysis of Discourse: The English Used by Teachers and Pupils*. London: Oxford University Press.

Sleeman, Derek H. and John S. Brown, eds. 1982. *Intelligent Tutoring Systems*. New York: Academic Press.

Smith, Ronnie W. 1992. "Integration of Domain Problem Solving with Natural Language Dialog: The Missing Axiom Theory." In *Applications of Artificial Intelligence X: Knowledge-Based Systems* (SPIE v. 1707), pp. 270–275.

Spitkovsky, John A. 1992. "Negative Acknowledgments in Natural Language Tutoring Systems," talk at Illinois Institute of Technology on July 22, 1992.

Stenström, Anna-Brita. 1994. *An Introduction to Spoken Interaction.* London: Longman.

Stevens, Albert L. and Allan Collins. 1977. "The Goal Structure of a Socratic Tutor," *Proceedings of the 1977 Annual Conference of the Association for Computing Machinery*, Seattle.

Stevens, Albert L., Allan Collins and Sarah E. Goldin. 1979. "Misconceptions in Students' Understanding," *International Journal of Man-Machine Studies* 11(1): 145–156. Reprinted in [Sleeman & Brown 1982, pp. 13–24].

Stevens, Albert and Bruce Roberts. 1983. "Quantitative and Qualitative Simulation in Computer Based Training," *Journal of Computer-Based Instruction* 10(1–2): 16–19.

Traum, David and Elizabeth Hinkelman. 1992. "Conversation Acts in Task-Oriented Spoken Dialogue," *Computational Intelligence* 8(3): 575–599.

Van Marcke, K. 1990. "A Generic Tutoring Environment," In Luigia C. Aiello, ed., *ECAI '90: Proceedings of the Ninth European Conference on Artificial Intelligence*, Stockholm. London: Pitman.

Weizenbaum, Joseph. 1967. "Contextual Understanding by Computers," *Communications of the ACM* 10(8): 474–480.

Weld, Daniel S. 1994. "An Introduction to Least Commitment Planning," *AI Magazine* 15(1–2).

Wenger, Etienne. 1987. *Artificial Intelligence and Tutoring Systems: Computational and Cognitive Approaches to the Communication of Knowledge*. Los Altos, CA: Morgan Kaufmann.

Wilkins, David E. 1988. *Practical Planning: Extending the Classical AI Planning Paradigm*. San Mateo, CA: Morgan Kaufmann.

Williams, Michael, James Hollan and Albert Stevens. 1981. "An Overview of STEAMER: An Advanced Computer-Assisted Instruction System for Propulsion Engineering," *Behavior Research Methods & Instrumentation* 13(2): 85–90.

Wong, Lung-Hsiang, Chee-Kit Looi and Hiok-Chai Quek. 1996. "Issues in Computerizing the Inquiry Dialogue Planning Process." In [Frasson, Gauthier & Lesgold 1996, pp. 252–260].

Woo, Chong W. 1991. *Instructional Planning in an Intelligent Tutoring System: Combining Global Lesson Plans with Local Discourse Control*. Ph.D. thesis, Department of Computer Science, Illinois Institute of Technology.

Woolf, Beverly P. 1984. *Context-Dependent Planning in a Machine Tutor*. Ph.D. thesis, Department of Computer Science, University of Massachusetts at Amherst. COINS Technical Report 84–21.

Woolf, Beverly P. 1987. "Representing Complex Knowledge in an Intelligent Machine Tutor," *Computational Intelligence* 3(1). Reprinted in John Self, ed., *Artificial Intelligence and Human Learning*, London: Chapman and Hall, 1988, pp. 3–27.

Young, R. Michael. 1994. *A Developer's Guide to the LONGBOW Discourse Planning System*. Technical Report 94–4, Intelligent Systems Program, University of Pittsburgh.

Young, R. Michael and Johanna D. Moore. 1994a. "Does Discourse Planning Require a Special-Purpose Planner?," *Proceedings of the AAAI Workshop on Planning for Inter-Agent Communication*, Seattle.

Young, R. Michael and Johanna D. Moore. 1994b. "DPOCL: A Principled Approach to Discourse Planning," *Proceedings of the Seventh International Workshop on Natural Language Generation*, Kennebunkport, ME, pp. 13–20.

Young, R. Michael, Johanna D. Moore and Martha E. Pollack. 1994. "Towards a Principled Representation of Discourse Plans," *Proceedings of the 16th Annual Conference of the Cognitive Science Society*, Atlanta. Hillsdale, NJ: Lawrence Erlbaum, pp. 946–951.

Young, R. Michael, Martha E. Pollack and Johanna D. Moore. 1994. "Decomposition and Causality in Partial Order Planning," *Proceedings of the Second International Conference on AI and Planning Systems*, Chicago, pp. 946–951.

Zhang, Yuemei. 1991. *Knowledge-Based Discourse Generation for an Intelligent Tutoring System*. Ph.D. thesis, Department of Computer Science, Illinois Institute of Technology.

Appendix A
Solutions to the Cardiovascular Problems

A.1 Table of variables

The following chart shows the abbreviations used in this Appendix and their meanings.

	Abbreviation	Variable
Core variables	CO	Cardiac output
	CVP	Central venous pressure
	HR	Heart rate
	IS	Inotropic state
	MAP	Mean arterial pressure
	SV	Stroke volume
	TPR	Total peripheral resistance
Other procedure variables	R_a	Arteriolar resistance
	BV	Blood volume
	P_{it}	Intrathoracic pressure
	VR	Venous return
	CBV	Central blood volume

A.2 Table of procedure variables and primary variables

The following chart shows the major procedures, the procedure variable or first variable to be affected, the primary variable or first core variable to be affected, and the direction of change. We refer to the primary variable as *clamped* when it cannot change further in RR (see Section 2.3.2).

Procedure	Procedure Variable	Primary Variable	Clamped?
Speed up artificial pacemaker	HR +	HR +	Yes
Slow down artificial pacemaker	HR –	HR –	
Give cholinergic antagonist (blocker)	HR +	HR +	No
Give cholinergic agonist	HR –	HR –	
Give beta-adrenergic antagonist	HR –, IS –	HR –, IS –	No/depends

Give beta-adrenergic agonist	HR +, IS +	HR +, IS +	
Increase R_a by 50%	R_a +	TPR +	No
Decrease R_a	R_a –	TPR –	
Give alpha-adrenergic agonist	TPR +	TPR +	Depends
Give alpha-adrenergic antagonist	TPR –	TPR –	
Transfusion	BV +	CVP +	N/A
Hemorrhage	BV –	CVP –	
Increase intrathoracic pressure (P_{it})	P_{it} +	CVP +	N/A
Decrease intrathoracic pressure (P_{it})	P_{it} –	CVP –	
Increase venous return (VR)	VR +	CVP +	N/A
Decrease venous return (VR)	VR –	CVP –	
Spin patient in centrifuge	CBV –	CVP –	N/A

Notes:

1) "Depends" means that the variable will be clamped if a sufficient quantity of the drug is administered. This information must be given in the statement of the problem. The problems are usually set up so that a sufficient quantity of a drug is administered to clamp the primary variable.

   The problem involving R_a is defined as reducing R_a by 50%. This is not sufficient to clamp TPR.

2) "N/A" refers to the fact that only neural variables can be clamped.

3) Beta-blockers have two primary variables. HR is not clamped. IS is clamped if a sufficient quantity of the drug is given. (See Section 2.4.5.)

A.3 Summary of rules

Determinants:

HR, TPR and IS are neural.
The determinants of SV are CVP and IS, and MAP is a minor determinant (inverse direction).
The determinants of CO are SV and HR.
The determinants of MAP are CO and TPR.
CO is the sole determinant of CVP (inverse direction).

Propagation-meta-rule:

Once a variable has been assigned a value in a stage, that value is never recomputed during that stage.

Determine-DR:

1. Determine the procedure variable and its direction from the perturbation.
2. Derive the primary variable and its direction from the procedure variable.
3. Determine the values of the neural variables.
4. Propagate values to variables on the shortest path to MAP.
5. Propagate values to the variables on the secondary path.

Determine-RR:

1. Determine values for the neural variables.
2. Propagate values to variables on the most important path to MAP.
3. Propagate values to variables on the secondary path.

Determine-SS:

1. Determine the value of the primary variable.
2. Determine values for the remaining neural variables.
3. Propagate values to variables on the shortest path to MAP.
4. Propagate values to variables on the secondary path.

Determine-shortest-path:

The shortest path is determined by the primary variable.

HR primary:	HR $\rightarrow$ CO $\rightarrow$ MAP
HR and IS primary:	HR $\rightarrow$ CO $\rightarrow$ MAP
No primary variable:	HR $\rightarrow$ CO $\rightarrow$ MAP
IS primary:	IS $\rightarrow$ SV $\rightarrow$ CO $\rightarrow$ MAP
TPR primary:	TPR $\rightarrow$ MAP
CVP primary:	CVP $\rightarrow$ SV $\rightarrow$ CO $\rightarrow$ MAP

Determine-secondary-path:

The secondary path is determined by the path which has already been traversed.

HR $\rightarrow$ CO $\rightarrow$ MAP:	CO $\rightarrow$ CVP $\rightarrow$ SV
IS $\rightarrow$ SV $\rightarrow$ CO $\rightarrow$ MAP:	CO $\rightarrow$ CVP
TPR $\rightarrow$ MAP:	MAP $\rightarrow$ SV $\rightarrow$ CO $\rightarrow$ CVP
CVP $\rightarrow$ SV $\rightarrow$ CO $\rightarrow$ MAP:	(none)

Determine-most-important-path:

No clamped variable:	HR → CO → MAP
IS clamped:	HR → CO → MAP
TPR clamped:	HR → CO → MAP
HR clamped:	IS → SV → CO → MAP
HR and IS clamped:	TPR → MAP

Det-procedure-variable:

When the value of the perturbation is propagated, the procedure variable is the first variable on the extended concept map to receive a value.

Det-primary-variable:

When the value of the procedure variable is propagated, the primary variable is the first core variable to receive a value.

Neural-DR:

Neural variables which are not primary do not change in DR.

Prop-1:

When a variable has one determinant, then the value of that determinant (increased, decreased or unchanged) determines the value of the variable.

Prop-1-Pit:

The following case is an exception to rule *Prop-1*. When intrathoracic pressure (P_{it}) is the primary variable, SV is inversely determined by CVP.

Prop-2n:

(no value) If a variable has two determinants and one of them has a value and the other one does not have a value yet, then the value of the one with a value determines the value of the variable.

Prop-2u:

(unchanged value) If a variable has two determinants and one of them has increased or decreased and the other one is unchanged, then the value of the one which has changed determines the value of the variable.

Prop-2e:

(equal values) If a variable has two determinants and both of them have the same value, then that value becomes the value of the variable.

Prop-2sv:

(conflicting values for SV) In DR, if the two determinants of SV (i.e. IS and CVP) have conflicting values (one increases and the other decreases), the value of CVP determines the value of SV.

Prop-2c:

(conflicting values for CO and MAP) If a variable other than SV has two determinants with conflicting values, and one determinant is neural and the other is not, then the non-neural determinant takes precedence unless the neural determinant is primary.

Prop-minor:

If a variable has two determinants and neither of them has a value other than "unchanged," but there is a minor determinant with a value, then the value of the minor determinant determines the value of the variable.

Neural-RR:

If MAP went up in DR, then any non-clamped neural variable will go down in RR, and vice versa.

Clamped-RR:

Clamped neural variables do not change in RR.

Algebraic-SS:

The value of a variable in SS is the "algebraic sum" of its qualitative predictions in DR and RR. If the DR and RR values have opposite signs, the DR value prevails.

Primary-SS:

If a variable is primary, it has the same value in SS as it did in DR.

Neural-SS:

If a neural variable is not primary, it has the same value in SS as it did in RR.

Denervate-DR:

If the baroreceptors have been denervated, then no variables change their values in DR.

Denervate-RR:

If the baroreceptors have been denervated, then all the neural variables rise in RR.

Clamped-RR-previous:

If a variable is clamped during the first perturbation of a multiple-perturbation problem, it does not change in RR of the second step except in the case where the variable in question is HR and the second procedure involves a broken pacemaker.

Denervate-RR-longterm:

If the first procedure involves denervating the baroreceptors, then no variables change in RR in the second procedure.

A.4 Solutions for the DR stage

A.4.1 DR: HR is the primary variable

Primary variable:

HR: Assume that the perturbation causes HR to rise.

Neural variables:

TPR: Since DR is by definition the stage before the baroreceptor reflex is activated, a neural variable can't change in the DR stage unless it's the primary variable. So TPR is unchanged.
(Neural-DR, V = TPR)

IS: By the same reasoning, IS is unchanged.
(Neural-DR, V = IS)

Shortest path to MAP:

CO: Since HR went up and SV has no value yet, CO (= HR * SV) must go up.
(Prop-2n, V = CO, $D_{changed}$ = HR, $D_{no\text{-}value}$ = SV)

MAP: CO going up will cause MAP to go up, since MAP = CO * TPR and TPR is unchanged.
(Prop-2u, V = MAP, $D_{changed}$ = CO, $D_{unchanged}$ = TPR)

Secondary path:

CVP: CO going up will cause CVP to go down.
(Prop-1, V = CVP, D = CO)

SV: Since CVP went down and IS is unchanged, SV will decrease.
(Prop-2u, V = SV, $D_{changed}$ = CVP, $D_{unchanged}$ = IS)

A.4.2 DR: TPR is the primary variable

Primary variable:

TPR: Assume that the perturbation causes TPR to rise.

Neural variables:

IS: Since DR is by definition the stage before the baroreceptor reflex is activated, a neural variable can't change in the DR stage unless it's the primary variable. So IS is unchanged.
(Neural-DR, V = IS)

HR: By the same reasoning, HR is unchanged.
(Neural-DR, V = HR)

Shortest path to MAP:

MAP: Since MAP = CO * TPR, CO has no value yet, and TPR has risen, MAP will rise.
(Prop-2n, V = MAP, $D_{changed}$ = TPR, $D_{no\text{-}value}$ = CO)

Secondary path:

SV: CVP has no value yet and IS is unchanged, so the minor determinant (MAP) comes into play. MAP going up will cause SV to decrease.
(Prop-minor, V = SV, D_1 = CVP, D_2 = SV, D_m = MAP)

CO: Since SV went down and HR is unchanged, CO will decrease.
(Prop-2u, V = CO, $D_{changed}$ = SV, $D_{unchanged}$ = HR)

CVP: Since CO decreased, CVP will increase.
(Prop-1, V = CVP, D = CO)

A.4.3 DR: IS is the primary variable

Primary variable:

IS: Assume that the perturbation causes IS to rise.

Neural variables:

HR: Since DR is by definition the stage before the baroreceptor reflex is activated, a neural variable can't change in the DR stage unless it's the primary variable. So HR is unchanged.
(Neural-DR, V = HR)

TPR: By the same reasoning, TPR is unchanged.
(Neural-DR, V = TPR)

Shortest path to MAP:

SV: Since IS went up and CVP has no value yet, SV will rise.
(Prop-2n, V = SV, $D_{changed}$ = IS, $D_{no\text{-}value}$ = CVP)

CO: Since SV went up and HR is unchanged, CO (= HR ∗ SV) will go up.
(Prop-2u, V = CO, $D_{changed}$ = SV, $D_{unchanged}$ = HR)

MAP: Since CO went up and TPR is unchanged, MAP (= CO ∗ TPR) will go up.
(Prop-2u, V = MAP, $D_{changed}$ = CO, $D_{unchanged}$ = TPR)

Secondary path:

CVP: CO going up will cause CVP to go down.
(Prop-1, V = CVP, D = CO)

A.4.4 DR: CVP is the primary variable

Primary variable:

CVP: Assume that the perturbation causes CVP to rise.

Neural variables:

IS: Since DR is by definition the stage before the baroreceptor reflex is activated, a neural variable can't change in the DR stage unless it's the primary variable. So IS is unchanged.
(Neural-DR, V = IS)

HR: By the same reasoning, HR is unchanged.
(Neural-DR, V = HR)

TPR: By the same reasoning, TPR is unchanged.
(Neural-DR, V = TPR)

Shortest path to MAP:

SV: Since CVP went up and IS is unchanged, SV will increase.
(Prop-2u, V = SV, $D_{changed}$ = CVP, $D_{unchanged}$ = IS)

CO: Since SV went up and HR is unchanged, CO (= HR * SV) must increase.
(Prop-2u, V = CO, $D_{changed}$ = SV, $D_{unchanged}$ = HR)

MAP: Since CO increased and TPR is unchanged, MAP (= CO * TPR) will go up.
(Prop-2u, V = MAP, $D_{changed}$ = CO, $D_{unchanged}$ = TPR)

Secondary path:

None.

A.5 Solutions for the RR stage

Without loss of generality, we assume that MAP increased in DR.

A.5.1 RR: No clamped variables

If the primary variable is not clamped, the solution for RR does not depend on which variable is primary.

Neural variables:

IS: Since MAP went up in DR, IS will decrease.
(Neural-RR, V = IS)

HR: By the same reasoning, HR will decrease.
(Neural-RR, V = HR)

TPR: By the same reasoning, TPR will decrease.
(Neural-RR, V = TPR)

Most important path to MAP:

CO: Since HR decreased and SV has no value yet, CO will go down.
(Prop-2n, V = CO, $D_{changed}$ = SV, $D_{no\text{-}value}$ = HR)

MAP: Since CO and TPR both decreased, MAP will go down.
(Prop-2e, V = MAP, D_1 = CO, D_2 = TPR)

Secondary path:

CVP: CO going down will cause CVP to go up.
(Prop-1, V = CVP, D = CO)

SV: Since IS decreased and CVP has increased, qualitative reasoning cannot decide. World knowledge tells us that SV will go up.
(Prop-2sv)

A.5.2 RR: HR is clamped

Neural variables:

IS: Since MAP went up in DR, IS will decrease.
(Neural-RR, V = IS)

HR: Since HR is clamped, it cannot change.
(Clamped-RR, V = HR)

TPR: Since MAP went up in DR, TPR will decrease.
(Neural-RR, V = TPR)

Most important path to MAP:

SV: Since IS decreased and CVP has no value yet, SV will go down.
(Prop-2n, V = SV, $D_{changed}$ = IS, $D_{no\text{-}value}$ = CVP)

CO: Since SV went down and HR is unchanged, CO will go down.
(Prop-2u, V = CO, $D_{changed}$ = SV, $D_{unchanged}$ = HR)

MAP: Since CO and TPR both decreased, MAP will go down.
(Prop-2e, V = MAP, D_1 = CO, D_2 = TPR)

Secondary path:

CVP: CO going down will cause CVP to go up because CO inversely affects CVP.
(Prop-1, V = CVP, D = CO)

A.5.3 RR: TPR is clamped

Neural variables:

IS: Since MAP went up in DR, IS will decrease.
(Neural-RR, V = IS)

HR: By the same reasoning, HR will decrease.
(Neural-RR, V = HR)

TPR: Since TPR is clamped, it cannot change.
(Clamped-RR, V = TPR)

Most important path to MAP:

CO: Since HR went down and SV has no value yet, CO will go down.
(Prop-2n, V = CO, $D_{changed}$ = HR, $D_{no\text{-}value}$ = SV)

MAP: Since CO decreased and TPR is unchanged, MAP will go down.
(Prop-2u, V = MAP, $D_{changed}$ = CO, $D_{unchanged}$ = TPR)

Secondary path:

CVP: CO going down will cause CVP to go up because CO inversely affects CVP.
(Prop-1, V = CVP, D = CO)

SV: Since IS decreased and CVP has increased, qualitative reasoning cannot decide. World knowledge tells us that SV will go up.
(Prop-2sv)

A.5.4 RR: IS is clamped

Neural variables:

IS: Since IS is clamped, it cannot change.
(Clamped-RR, V = IS)

HR: By the same reasoning, HR will decrease.
(Neural-RR, V = HR)

TPR: By the same reasoning, TPR will decrease.
(Neural-RR, V = TPR)

Most important path to MAP:

CO: Since SV has no value yet and HR decreased, CO will go down.
(Prop-2n, V = CO, $D_{changed}$ = HR, $D_{no\text{-}value}$ = SV)

MAP: Since CO and TPR both decreased, MAP will go down.
(Prop-2e, V = MAP, D_1 = CO, D_2 = TPR)

Secondary path:

CVP: CO going down will cause CVP to go up because CO inversely affects CVP.
(Prop-1, V = CVP, D = CO)

SV: Since CVP increased and IS is unchanged, SV will increase.
(Prop-2u, V = SV, $D_{changed}$ = CVP, $D_{unchanged}$ = IS)

A.6 Solutions for the SS stage

In this section, we give solutions using causal reasoning. Solutions using the algebraic method can easily be determined using the rules given in Section 2.4.4.

A.6.1 SS: HR is the primary variable

Neural variables:

HR: Since HR is primary, it has the same value in SS as in DR, so it rises.
(Primary-SS, V = HR)

TPR: Since TPR is not primary, it has the same value in SS as in RR, so it decreases.
(Neural-SS, V = TPR)

IS: Since IS is not primary, it has the same value in SS as in RR, so it decreases.
(Neural-SS, V = IS)

Shortest path to MAP:

CO: Since HR went up and SV has no value yet, CO (= HR * SV) must go up.
(Prop-2n, V = CO, $D_{changed}$ = HR, $D_{no\text{-}value}$ = SV)

MAP: CO went up and TPR went down. CO overrides TPR because TPR isn't primary, so MAP goes up.
(Prop-2c, V = MAP, $D_{non\text{-}neural}$ = CO, D_{neural} = TPR)

Secondary path:

CVP: CO going up will cause CVP to go down.
(Prop-1, V = CVP, D = CO)

SV: Since both CVP and IS went down, SV will decrease.
(Prop-2e, V = SV, D_1 = CVP, D_2 = IS)

A.6.2 SS: TPR is the primary variable

Primary variable:

TPR: Since TPR is primary, it has the same value in SS as in DR, so it rises.
(Primary-SS, V = TPR)

Neural variables:

IS: Since IS is not primary, it has the same value in SS as in RR, so it decreases.
(Neural-SS, V = IS)

HR: Since HR is not primary, it has the same value in SS as in RR, so it decreases.
(Neural-SS, V = HR)

Shortest path to MAP:

MAP: Since MAP = CO * TPR, CO has no value yet, and TPR has risen, MAP will rise.
(Prop-2n, V = MAP, $D_{changed}$ = TPR, $D_{no\text{-}value}$ = CO)

Secondary path:

SV: IS decreased and CVP has no value yet, so SV will decrease.
(Prop-2n, V = SV, $D_{changed}$ = IS, $D_{no\text{-}value}$ = CVP)

CO: Since SV and HR both went down, CO will decrease.
(Prop-2e, V = CO, D_1 = SV, D_2 = HR)

CVP: Since CO decreased, CVP will increase.
(Prop-1, V = CVP, D = CO)

A.6.3 SS: IS is the primary variable

Neural variables:

IS: Since IS is primary, it has the same value in SS as in DR, so it rises.
(Primary-SS, V = IS)

HR: Since HR is not primary, it has the same value in SS as in RR, so it decreases.
(Neural-SS, V = HR)

TPR: Since TPR is not primary, it has the same value in SS as in RR, so it decreases.
(Neural-SS, V = TPR)

Shortest path to MAP:

SV: Since IS went up and CVP has no value yet, SV will rise.
(Prop-2n, V = SV, $D_{changed}$ = IS, $D_{no\text{-}value}$ = CVP)

CO: SV went up and HR went down. SV overrides HR because HR isn't primary, so CO will go up.
(Prop-2c, V = CO, $D_{non\text{-}neural}$ = SV, D_{neural} = HR)

MAP: CO went up and TPR went down. CO overrides TPR because TPR isn't primary, so MAP will go up.
(Prop-2c, V = MAP, $D_{non\text{-}neural}$ = CO, D_{neural} = TPR)

Secondary path:

CVP: CO going up will cause CVP to go down.
(Prop-1, V = CVP, D = CO)

A.6.4 SS: CVP is the primary variable

Primary variable:

CVP: Since CVP is primary, it has the same value in SS as in DR, so it rises.
(Primary-SS, V = CVP)

Neural variables:

IS: Since IS is not primary, it has the same value in SS as in RR, so it decreases.
(Neural-SS, V = IS)

HR: Since HR is not primary, it has the same value in SS as in RR, so it decreases.
(Neural-SS, V = HR)

TPR: Since TPR is not primary, it has the same value in SS as in RR, so it decreases.
(Neural-SS, V = TPR)

Shortest path to MAP:

SV: Since IS decreased and CVP has increased, qualitative reasoning cannot decide. World knowledge tells us that SV will go up.
(Prop-2sv)

CO: SV went up and HR went down. SV overrides because HR isn't primary, so CO increases.
(Prop-2c, V = CO, $D_{non\text{-}neural}$ = SV, D_{neural} = HR)

MAP: CO increased and TPR decreased. CO overrides because TPR isn't primary, so MAP goes up.
(Prop-2c, V = MAP, $D_{non\text{-}neural}$ = CO, D_{neural} = TPR)

Secondary path:

None.

A.7 Prediction tables for the simple cases

This section summarizes the results for each case. The left prediction table refers to the case where the primary variable increases and the one on the right to the case where the primary variable decreases.

A.7.1 HR is primary, non-clamped

Parameter	DR	RR	SS
CVP	–	+	–
IS	0	–	–
SV	–	+	–
HR	+	–	+
CO	+	–	+
TPR	0	–	–
MAP	+	–	+

Parameter	DR	RR	SS
CVP	+	–	+
IS	0	+	+
SV	+	–	+
HR	–	+	–
CO	–	+	–
TPR	0	+	+
MAP	–	+	–

A.7.2 HR is primary, clamped

Parameter	DR	RR	SS
CVP	–	+	–
IS	0	–	–
SV	–	–	–
HR	+	0	+
CO	+	–	+
TPR	0	–	–
MAP	+	–	+

Parameter	DR	RR	SS
CVP	+	–	+
IS	0	+	+
SV	+	+	+
HR	–	0	–
CO	–	+	–
TPR	0	+	+
MAP	–	+	–

A.7.3 TPR is primary, non-clamped

Parameter	DR	RR	SS
CVP	+	+	+
IS	0	–	–
SV	–	+	–
HR	0	–	–
CO	–	–	–
TPR	+	–	+
MAP	+	–	+

Parameter	DR	RR	SS
CVP	–	–	–
IS	0	+	+
SV	+	–	+
HR	0	+	+
CO	+	+	+
TPR	–	+	–
MAP	–	+	–

A.7.4 TPR is primary, clamped

Parameter	DR	RR	SS
CVP	+	+	+
IS	0	–	–
SV	–	+	–
HR	0	–	–
CO	–	–	–
TPR	+	0	+
MAP	+	–	+

Parameter	DR	RR	SS
CVP	–	–	–
IS	0	+	+
SV	+	–	+
HR	0	+	+
CO	+	+	+
TPR	–	0	–
MAP	–	+	–

A.7.5 IS is primary, non-clamped

Parameter	DR	RR	SS
CVP	–	+	–
IS	+	–	+
SV	+	+	+
HR	0	–	–
CO	+	–	+
TPR	0	–	–
MAP	+	–	+

Parameter	DR	RR	SS
CVP	+	–	+
IS	–	+	–
SV	–	–	–
HR	0	+	+
CO	–	+	–
TPR	0	+	+
MAP	–	+	–

A.7.6 IS is primary, clamped

Parameter	DR	RR	SS
CVP	–	+	–
IS	+	0	+
SV	+	+	+
HR	0	–	–
CO	+	–	+
TPR	0	–	–
MAP	+	–	+

Parameter	DR	RR	SS
CVP	+	–	+
IS	–	0	–
SV	–	–	–
HR	0	+	+
CO	–	+	–
TPR	0	+	+
MAP	–	+	–

A.7.7 CVP is primary

Parameter	DR	RR	SS
CVP	+	+	+
IS	0	–	–
SV	+	+	+
HR	0	–	–
CO	+	–	+
TPR	0	–	–
MAP	+	–	+

Parameter	DR	RR	SS
CVP	–	–	–
IS	0	+	+
SV	–	–	–
HR	0	+	+
CO	–	+	–
TPR	0	+	+
MAP	–	+	–

A.8 Special cases

A.8.1 Beta-blockers

Beta-blockers have two primary variables, HR and IS. The sino-atrial node, which controls heart rate, contains both beta receptors and cholinergic receptors. Since beta-blockers block only the effect of the nervous system on beta receptors (i.e. through the sympathetic nervous system) and not on cholinergic receptors (i.e. through the parasympathetic nervous system), HR will not be clamped. IS will be clamped if a sufficient quantity of the drug is given.

As a result, the result of administering a beta-agonist or a beta-blocker is the same as the result given for HR primary and non-clamped in A.7.1 except for IS:

Parameter	DR	RR	SS
CVP	–	+	–
IS	+	0	+
SV	–	+	–
HR	+	–	+
CO	+	–	+
TPR	0	–	–
MAP	+	–	+

Parameter	DR	RR	SS
CVP	+	–	+
IS	–	0	–
SV	+	–	+
HR	–	+	–
CO	–	+	–
TPR	0	+	+
MAP	–	+	–

The logic for deriving these values is as given above for HR primary and non-clamped (A.4.1, A.5.1, and A.6.1) with the exception of IS and SV. For SV, although the logic for deriving the result is different, the result is the same.

DR:

IS: IS is a primary variable; its direction of change depends on the perturbation. Since IS and HR change in the same direction, we assume IS increases here because HR increases in A.4.1.)

SV: Since IS increased and CVP decreased, qualitative reasoning cannot decide. World knowledge tells us that SV will go down. *(Prop-2sv)*

RR:

IS: Since IS is clamped, it cannot change. *(Clamped-RR, V = IS)*

SV: Since CVP increased and IS is unchanged, SV will increase. *(Prop-2u, V = SV, $D_{changed}$ = CVP, $D_{unchanged}$ = IS)*

SS:

IS: Since IS is primary, it has the same value in SS as in DR, so it rises.
(Primary-SS, V = IS)

SV: Since IS increased and CVP has decreased, qualitative reasoning cannot decide. World knowledge tells us that SV will go down.
(Prop-2sv)

A.8.2 Changing intrathoracic pressure (P_{it})

When intrathoracic pressure (P_{it}) increases, all of the structures inside the chest are compressed, including the central veins and the chambers of the heart. Since the central veins are compressed, pressure inside them rises, so the primary variable is central venous pressure (CVP), which increases. Normally an increase in CVP would cause more blood to flow into the heart on each beat, i.e. an increase in ventricular filling. This would cause more blood to flow out per beat also, i.e. stroke volume (SV) would increase. But when P_{it} increases, the arteries where the blood will go on leaving the heart are compressed also. As a result, ventricular filling decreases, causing a decrease in SV.

Since this difference is propagated through the other variables, one obtains the opposite of the normal case for many variables.

Parameter	DR	RR	SS
CVP	+	–	+
IS	0	+	+
SV	–	–	–
HR	0	+	+
CO	–	+	–
TPR	0	+	+
MAP	–	+	–

Parameter	DR	RR	SS
CVP	–	+	–
IS	0	–	–
SV	+	+	+
HR	0	–	–
CO	+	–	+
TPR	0	–	–
MAP	+	–	+

DR:

Primary variable:

CVP: Primary variable: assume that the perturbation causes CVP to rise.

Neural variables:

IS, HR, TPR: Since DR is by definition the stage before the baroreceptor reflex is activated, a neural variable can't change in the DR stage unless it's the primary variable. So all of these variables are unchanged.
(Neural-DR, IS/HR/TPR)

Shortest path to MAP:

SV: Although an increase in CVP would normally cause an increase in SV, SV decreases because the compression of the heart causes a decrease in ventricular filling (see Section 2.4.5).
(Prop-Pit)

CO: Since SV went down and HR is unchanged, CO (= HR * SV) must decrease.
(Prop-2u, V = CO, $D_{changed}$ = SV, $D_{unchanged}$ = HR)

MAP: Since CO decreased and TPR is unchanged, MAP (= CO * TPR) will go down.
(Prop-2u, V = MAP, $D_{changed}$ = CO, $D_{unchanged}$ = TPR)

RR:

Neural variables:

IS, HR, TPR: Since MAP went down in DR, all of the neural variables will increase.
(Neural-RR, V = IS/HR/TPR)

Most important path to MAP:

CO: Since HR increased and SV has no value yet, CO will go up.
(Prop-2n, V = CO, $D_{changed}$ = HR, $D_{no\text{-}value}$ = SV)

MAP: Since CO and TPR both increased, MAP will go up.
(Prop-2e, V = MAP, D_1 = CO, D_2 = TPR)

Secondary path:

CVP: CO going up will cause CVP to go down.
(Prop-1, V = CVP, D = CO)

SV: Since IS increased and CVP has decreased, qualitative reasoning cannot decide. World knowledge tells us that SV will go down.
(Prop-2sv)

SS:

Primary variable:

CVP: Since CVP is primary, it has the same value in SS as in DR, so it rises.
(Primary-SS, V = CVP)

Neural variables:

IS, HR, TPR: Since the neural variables are not primary, they have the same value in SS as in RR, so they increase.
(Neural-SS, V = IS/HR/TPR)

Shortest path to MAP:

SV: Although an increase in CVP would normally cause an increase in SV, SV decreases as explained for DR.
(Prop-Pit)

CO: SV went down and HR went up. SV overrides because HR isn't primary, so CO decreases.
(Prop-2c, V = CO, $D_{non\text{-}neural}$ = SV, D_{neural} = HR)

MAP: CO decreased and TPR increased. CO overrides because TPR isn't primary, so MAP goes down.
(Prop-2c, V = MAP, $D_{non\text{-}neural}$ = CO, D_{neural} = TPR)

A.8.3 Denervating the baroreceptors

To enervate the baroreceptors means to cut the link from the baroreceptors to the nervous system. Since the nervous system is not active in DR, this means that the effect of the perturbation is not noticed in DR, i.e. no variables change. In RR, the complete

cessation of signals from the nervous system will appear to the receptors as a very steep fall in MAP. As a result all of the neural variables will rise in RR, and none is clamped.

Parameter	DR	RR	SS
CVP	0	–	–
IS	0	+	+
SV	0	–	–
HR	0	+	+
CO	0	+	+
TPR	0	+	+
MAP	0	+	+

DR:

All variables: Since DR is by definition the stage before the baroreceptor reflex is activated, disabling this effect will not be noticed in DR, i.e. no variables will change.
(Denervate-DR, V = all)

RR:

Neural variables:

IS, HR, TPR: The complete cessation of signals from the nervous system will appear to the receptors as a very steep fall in MAP. As a result all of the neural variables will rise in RR.
(Denervate-RR, V = IS/HR/TPR)

Since there is no clamped variable, the remainder of RR follows A.5.1. The signs are reversed because MAP rises in A.5.1.

Most important path to MAP:

CO: Since HR increased and SV has no value yet, CO will go up.
(Prop-2n, V = CO, $D_{changed}$ = HR, $D_{no\text{-}value}$ = SV)

MAP: Since CO and TPR both increased, MAP will go up.
(Prop-2e, V = MAP, D_1 = CO, D_2 = TPR)

Secondary path:

CVP: CO going up will cause CVP to go down.
(Prop-1, V = CVP, D = CO)

SV: Since IS increased and CVP has decreased, qualitative reasoning cannot decide. World knowledge tells us that SV will go down.
(Prop-2sv)

SS:

Neural variables:

IS, HR, TPR: Since none of these variables is primary, they have the same value in SS as in RR, so they rise.
(Neural-SS, V = IS/HR/TPR)

Shortest path to MAP:

CO: Since HR increased and SV has no value yet, CO will go up.
(Prop-2n, V = CO, $D_{changed}$ = HR, $D_{no\text{-}value}$ = SV)

MAP: Since CO and TPR both increased, MAP will go up.
(Prop-2e, V = MAP, D_1 = CO, D_2 = TPR)

Secondary path:

CVP: CO going up will cause CVP to go down.
(Prop-1, V = CVP, D = CO)

SV: Since IS increased and CVP has decreased, qualitative reasoning cannot decide. World knowledge tells us that SV will go down.
(Prop-2sv)

A.8.4 Compound case: Beta-blocker, then broken pacemaker

When two perturbations occur in sequence, we process them sequentially and illustrate the result by showing the prediction table after each perturbation.

The DR for the second procedure is based on the result of SS in the first procedure. In other words, DR for the second procedure measures the change from the previous steady state. This result differs from the regular DR in only one case. When the first

perturbation causes a variable to be clamped, it stays clamped in RR during the second procedure, with the exception that a broken pacemaker in the second procedure can override the result of clamping by a drug.

When a beta-blocker is followed by a broken pacemaker, the first prediction table is the right-hand table in A.8.1 (beta-blockers). Here is the second table, showing the result when the pacemaker increases or decreases, respectively. The logic for constructing this table is shown below.

Parameter	DR	RR	SS
CVP	–	–	–
IS	0	0	0
SV	–	+	–
HR	+	0	+
CO	+	+	+
TPR	0	–	–
MAP	+	–	+

Parameter	DR	RR	SS
CVP	+	+	+
IS	0	0	0
SV	+	–	+
HR	–	0	–
CO	–	–	–
TPR	0	+	+
MAP	–	+	–

DR:

Logic is the same as A.4.1 (DR for HR primary). (Also, note that IS does not change in any case, as it is still clamped.)

RR:

Neural variables:

IS: Since IS is clamped by the beta-blocker, it cannot change.
(Clamped-RR-previous, V = IS)

HR: Since HR is clamped by the broken pacemaker, it cannot change.
(Clamped-RR, V = HR)

TPR: Since MAP went up in DR, TPR will decrease.
(Neural-RR, V = TPR)

Most important path to MAP:

MAP: Since MAP = CO * TPR, CO has no value yet, and TPR went down, MAP will go down.
(Prop-2n, V = MAP, $D_{changed}$ = TPR, $D_{no\text{-}value}$ = CO)

Secondary path:

SV: CVP has no value yet and IS is unchanged, so the minor determinant (MAP) comes into play. MAP going down will cause SV to increase.
(Prop-minor, V = SV, D_1 = CVP, D_2 = SV, D_m = MAP)

CO: Since SV went up and HR is unchanged, CO will increase.
(Prop-2u, V = CO, $D_{changed}$ = SV, $D_{unchanged}$ = HR)

CVP: Since CO went up, CVP will go down.
(Prop-1, V = CVP, D = CO)

SS:

Neural variables:

HR: Since HR is primary, it has the same value in SS as in DR, so it rises.
(Primary-SS, V = HR)

TPR: Since TPR is not primary, it has the same value in SS as in RR, so it decreases.
(Neural-SS, V = TPR)

IS: Since IS is not primary, it has the same value in SS as in RR, so it is unchanged. (Note that IS could not change in any case, as it is still clamped.)
(Neural-SS, V = IS)

Shortest path to MAP:

CO: Since HR went up and SV has no value yet, CO (= HR * SV) must go up.
(Prop-2n, V = CO, $D_{changed}$ = HR, $D_{no\text{-}value}$ = SV)

MAP: CO went up and TPR went down. CO overrides TPR because TPR isn't primary, so MAP goes up.
(Prop-2c, V = MAP, $D_{non\text{-}neural}$ = CO, D_{neural} = TPR)

Secondary path:

CVP: CO going up will cause CVP to go down.
(Prop-1, V = CVP, D = CO)

SV: Since CVP went down and IS is unchanged, SV will decrease.
(Prop-2u, V = SV, $D_{changed}$ = CVP, $D_{unchanged}$ = IS)

A.8.5 Compound case: Denervate the baroreceptors, then beta-blocker

When denervation of the baroreceptors is followed by a beta-blocker, the first prediction table is given in A.8.3 (denervating the baroreceptors). The second prediction table and the logic for deriving it are shown below.

Parameter	DR	RR	SS
CVP	+	0	+
IS	–	0	–
SV	+	0	+
HR	–	0	–
CO	–	0	–
TPR	0	0	0
MAP	–	0	–

DR:

Same as DR in A.8.1 (beta-blocker). The signs are reversed because the logic in A.8.1 shows the effect of a beta-agonist.

RR:

All variables: Since the nervous system receives no signals from the baroreceptors, it cannot pass any on. Thus no variables change.
(Denervate-RR-longterm, V = all)

SS:

Neural variables:

IS: Since IS is primary, it has the same value in SS as in DR, so it decreases.
(Primary-SS, V = IS)

HR: Since HR is primary, it has the same value in SS as in DR, so it decreases.
(Primary-SS, V = HR)

TPR: Since TPR is not primary, it has the same value in SS as in RR, so it is unchanged.
(Neural-SS, V = TPR)

Shortest path to MAP:

CO: Since HR went down and SV has no value yet, CO (= HR * SV) must go down.
(Prop-2n, V = CO, $D_{changed}$ = HR, $D_{no\text{-}value}$ = SV)

MAP: CO went down and TPR is unchanged, so MAP goes down.
(Prop-2u, V = MAP, $D_{changed}$ = CO, $D_{unchanged}$ = TPR)

Secondary path:

CVP: CO going down will cause CVP to go up.
(Prop-1, V = CVP, D = CO)

SV: Since IS decreased and CVP has increased, qualitative reasoning cannot decide. World knowledge tells us that SV will go up.
(Prop-2sv)